

Ivica Kartelo

Metodika

podučavanja
programiranja
PHP5 MySQL

Škola E-92 d.o.o. Split

Naslov: Metodika podučavanja programiranja PHP5 MySQL

Autor: Ivica Kartelo

Urednik: Ivica kartelo

Nakladnik: Škola E92 d.o.o. Split www.e92.hr

Za nakladnika: Ivica Kartelo

Tisak: Kartular d.o.o. Split

Prijelom teksta: Ivica Kartelo

Grafički dizajn naslovnica: Nebojša Bošnjak

U ovoj knjizi pojavljuje se nekoliko zaštićenih znakova i naziva čija je uporaba ograničena zakonom. Popis tih znakova i naziva kao i njihovih vlasnika nalazi se kod nakladnika.

© Ivica Kartelo

Sva imena firmi, osoba, proizvoda i usluga koja se pojavljuju u primjerima su izmišljena.

Autor, Nakladnik i Tiskar ne snose odgovornost za posljedice korištenja ove knjige, niti za pogreške nastale u procesu stvaranja knjige.

Sadržaj

U V O D U P R O G R A M I R A N J E	1 4
Od Aristotela do Boolea i Gatesa	14
N a s t a v n a c j e l i n a	1
PHP za početnike	13
Metodika	13
Mogući problemi prilikom instalacije PHP	13
Prva aplikacija	13
Instrukcija	13
HTML (XHTML) i CSS	14
Druga PHP aplikacija	14
Ističem!	14
Što istaknuti u ovoj početnoj fazi	16
Varijable	16
Lokalne i globalne varijable	17
Forma	19
IF nas je sve prevario	19
Ističem kôd IF	20
Petlje	21
Moja iskustva	21
Vježbe	21
Teorija (Diskusija)	23
Primjer	23
Usmeni	23
Primjer	24
Pismeni	24
Domaći rad	24
Email	24
Udžbenik	25
Što ćemo naučiti u prvoj nastavnoj cjelini?	25
Instalacija PHP	25
Zaustavite rad servisa IIS Admin	25
Instalacija Apachea i PHP	25
Moje prve programske instrukcije u PHP-u	26
Prva PHP aplikacija	26
Pogledajmo u radu našu prvu PHP aplikaciju	26
Objašnjenje moje prve PHP aplikacije	27
Moja druga PHP aplikacija	27
Objašnjenje moje druge PHP aplikacije	28
Zadaci	29

Test	30
PHP varijable	31
Tipovi PHP varijabli	31
Varijable $1 + 2 = 3$	31
Objašnjenje $1 + 2 = 3$	32
Koliko žive varijable, a koliko živi program	33
Objašnjenje	34
Globalne varijable	34
Lokalne varijable	35
Dohvat globalne varijable iz funkcije	36
Forma	36
Varijable izvan PHP	37
Objašnjenje	37
Varijable: Zadaci	38
Varijable: Test	39
PHP uvjetni programski kôd	40
IF...ELSE	40
Objašnjenje	40
Zadaci	41
Test	41
Logički i usporedni operatori	41
Tablica istine za logičko 'i'	42
Tablica istine za logičko 'ili'	42
Zadaci iz logičkih i usporednih operatora	42
Objašnjenje logičkih operatora u gornjim primjerima	44
Petlje	45
For	45
Objašnjenje	45
FOR: Zadaci	45
Test	45
While	46
Preporuka profesorima i učenicima	48
Zaključak nastavne cjeline 1	48
Na redu je pristup bazi podataka. Ali prije toga moramo se upoznati s najčešćim poslužiteljem baza podataka koji se danas koristi uz PHP – MySQL-om.	48
N a s t a v n a c j e l i n a 2 MySQL za početnike	50
Metodika -----	50
Instalacija	50
Vježbe	50
phpMyAdmin	50

Udžbenik-----	51
Što ćemo naučiti u ovoj nastavnoj cjelini? 51	
Instalacija MySQL 51	
Što je točno MySQL? 51	
Naša komunikacija s MySQL 51	
MySQL monitor 52	
Pokretanje MySQL monitora preko Command Prompt 52	
Objašnjenje inicijalnih korisničkih imena MySQL-a 53	
Kreiranje baze podataka 'ulazni_racuni' 56	
Kreiranje tablica 56	
Unos podataka u tablice 57	
Pregled tablica 58	
Ispravljanje datuma u tablici racuni 59	
Brisanje baze. Brisanje tablica 60	
Promjene u strukturi tablice 60	
Primarni ključ 61	
Auto_increment 62	
SQL naredba SELECT 64	
Grupiranje redaka po nekom zajedničkom podatku (Group by) 64	
Group by 66	
MySQL funkcije 66	
Zadaci 66	
Test 67	
Zaključak 68	
N a s t a v n a c j e l i n a 3 PHP pristup MySQL bazi 70	
Metodika-----	70
Udžbenik-----	71
Napravimo bazu ulazni_racuni 71	
Objašnjenje 73	
PHP kao korisnik 73	
Sjetimo se Command Prompta i MySQL monitora 73	
PHP prolazi isti gore opisani postupak 73	
Kako se razumiju PHP i MySQL 74	
Vidimo što nam je baza vratila 74	
Objašnjenje 75	
A sad želimo vidjeti tablični prikaz 76	
Zadaci 77	
Rješenja 77	
Zadaci 80	

Test 81

N a s t a v n a c j e l i n a 4 : : P H P B l o g 83

Metodika----- 83

Udžbenik----- 84

Što ćemo napraviti? 84

Baza 'blog' 84

 Blog - PHP stranice 84

 index.php 84

 dodaj.php 85

 dodaj_pohrani.php 85

 admin.php 86

 delete.php 87

 edit.php 87

 edit_pohrani.php 88

Objašnjenje 89

 index.php 89

 dodaj.php 91

 dodaj_pohrani.php 93

 GET i POST 94

 Nastavak .php ili .html 94

 admin.php 94

 edit.php 95

 edit_pohrani.php 96

 delete.php 97

Auto_increment primary key index 97

Zadaci 98

Test 99

N a s t a v n a c j e l i n a 5 P H P L o g i n 101

Metodika----- 101

Udžbenik----- 101

Što ćemo naučiti? 101

Sesija 102

 loginForm.php 103

 login.php 103

 sigurnosniKod.php 103

 odjava.php 104

 tajne_konstante.php 104

 <?php require_once("sigurnosniKod.php");?> 105

 Objašnjenje 105

loginForma.php	105
login.php	106
sigurnosniKod.php	108
odjava.php	109
tajne_konstante.php	111
Zaključak	111
Zadaci	111
Test	112
N a s t a v n a c j e l i n a 6	PHP Objektno orijentirano programiranje 113
Metodika-----	113
Udžbenik-----	115
Što ćemo naučiti?	115
Klasa	115
Moja prva klasa	116
Objašnjenje klase	116
Svojstva (properties) klase (ili članovi klase, ili varijable klase)	117
Metode	117
\$this	117
Objekt	118
Moj prvi objekt	118
Objašnjenje objekta	118
Instanciranje objekta	118
Inicijaliziranje objekta	118
Objekt poziva svojstvo (properties)	118
Objekt poziva metodu	119
Objašnjenje	119
Konstruktor	119
student.php	119
studenti.php	120
Objašnjenje	120
student.php	120
studenti.php	120
Kad otvorimo studenti.php, vidjet ćemo	121
Objašnjenje	121
Klasa nasljednik	121
studentnasljednik.php	121
studenti_nasljednici.php	121
Objašnjenje	122
studentnasljednik.php	122

studenti_naljednici.php	122
Prijenos vrijednosti preko parametara klase u varijable konstruktora	123
Prijenos vrijednosti preko parametara funkcije u varijable funkcije	123
Zaključak	123
Ovo će biti prikazano na web stranici kad otvorimo stranicu studenti_nasljednici.php	123
Objašnjenje	124
Operator ::	124
Dohvat konstante izvana bez instanciranja klase	124
split.php	124
split2.php	124
Kad otvorimo stranicu split2.php vidjet ćemo	125
Objašnjenje	125
Dohvat statičkih svojstava i statičkih metoda iznutra i izvana bez instanciranja klase	125
split1.php	125
Na stranici će biti prikazano ovo	125
Objašnjenje	125
split1.php	125
Kreiramo statičko svojstvo.	125
Kreiramo statičku metodu	126
Pristupamo IZNUTRA konstanti roditelja	126
Pristupamo iznutra svom statičkom svojstvu	126
Pristupamo izvana statičkoj metodi bez instanciranja objekta	126
Na web stranici će biti prikazano ovo	126
Zadaci	126
Rješenje zadataka	128
Test	130
Odgovori na test	131
N a s t a v n a c j e l i n a 7 PHP Smarty	134
Metodika-----	134
Udžbenik-----	134
Što ćemo naučiti?	134
Razdvajanje XHTML od PHP	135
Instalacija vježbi	135
Cilj vježbi	135
Vježba 1	135
XHTML i PHP zajedno u istom dokumentu	135
Vježba 2	136
XHTML i PHP razdvojeni	136
Alias	136

Objašnjenje	138
templates_c	138
index.php	138
index.tpl	139
Ključne riječi assign i display	139
Zaključak	139
Vježba 3	140
Sadržaj iz baze podataka bez smartya	140
Vježba 4	141
Podaci iz baze podataka prikazani pomoću Smartya	141
index.tpl	141
index.php	141
Objašnjenje	142
templates_c	142
Vježba 5	142
Više sadržaja	142
index.tpl	143
index.php	143
Objašnjenje	144
Vježba 6	144
zaglavlje.tpl	144
index.tpl	145
podnozje.tpl	145
index.php	145
templates_c	145
Objašnjenje index.tpl	145
Vježba 7	146
array	146
index.php	146
index.tpl	146
Web stranica će prikazati ovo	146
Vježba 8	146
Asocijativni array	147
index.php	147
index.tpl	147
Web stranica	148
Objašnjenje	148
Zaključak	149
N a s t a v n a c j e l i n a 8 PHP SMARTY i PEAR	151

Metodika-----	151
Ovdje je naglasak na 153	
Smarty 153	
Pear 153	
Aplikacija u tri sloja 153	
Udžbenik-----	154
Što ćemo naučiti? 154	
Pripreme za vježbe 154	
Alias 154	
Mape za vježbe 155	
Vježba 1 155	
konfiguracija.inc.php 155	
uspostavljanje_smarty.php 155	
vrh_aplikacije.php 155	
U mapi CONFIGS kreirajte ovaj dokument 156	
web_site.conf 156	
U mapu BIBLIOTEKE ide mapa Smarty 156	
U mapi TEMPLATES kreirajte ove dokumente 156	
index.tpl 156	
zaglavlje.tpl 156	
U mapi k1 kreirajte 156	
Web stranica će prikazati ovo 156	
templates_c 157	
Objašnjenje 157	
konfiguracija.inc.php 157	
uspostavljanje_smarty.php 158	
vrh_aplikacije.php 159	
Mapa k1 159	
index.php 159	
Mapa templates 159	
index.tpl 159	
Zaključak 160	
Vježba 2 161	
Priprema za vježbu 2 161	
PEAR 161	
Instalacija PEAR-a 161	
Kreiranje baze podataka 'knjigagostiju' 161	
Što će nam prikazati web stranica na kraju ove vježbe 162	
Projektiranje web aplikacije u tri sloja 163	

- Podatkovni sloj 163
 - podatkovni_objekt.php 163
- Poslovni sloj 164
 - poslovni_objekt.php 164
- Prezentacijski sloj 164
 - knjiga_gostiju.tpl 164
 - function.load_knjiga_gostiju.php 165
 - index.tpl 165
 - zaglavlje.tpl 166
- Ostale potrebne nove datoteke i promjene na postojećim datotekama 166
 - index.php 166
 - vrh_aplikacije.php 166
 - uspostavljanje_smarty.php 166
 - konfiguracija.inc.php 167
 - dno_aplikacije.php 167
 - bazapodataka.php 167
 - Otvorite u web pregledniku aplikaciju Knjiga gostiju 168
- Objašnjenje 168
 - Put kojim je sadržaj stigao iz baze do web stranice 169
 - index.php 169
 - vrh_aplikacije.php 169
 - konfiguracija.inc.php: 170
 - uspostavljanje_smarty.php: 171
 - bazapodataka.php.php: 172
 - Inicijaliziranje prvog objekta 175
 - Podsjetimo se do kud smo stigli u procesuiranju naše aplikacije 175
 - poslovni_objekt.php 175
 - podatkovni_objekt.php 176
 - poslovni_objekt.php NASTAVAK 177
 - Na ovom mjestu procesuiranja naše aplikacije 177
 - Gdje smo stigli? 177
 - Nastavak objašnjenja 178
 - index.php 178
 - index.tpl 178
 - Poziv {include file="knjiga_gostiju.tpl"} pokrenuo je procesuiranje prezentacijskog sloja 179
 - function.load_knjiga_gostiju.php 179
 - function.load_knjiga_gostiju.php – NASTAVAK 183
 - knjiga_gostiju.tpl 183
 - Gdje smo stigli? 184

dno_aplikacije.php	185
Kraj procesuiranja	186
Sve instrukcije naše aplikacije na jednom mjestu	186
Runtime aplikacije	190
Sažetak runtimea aplikacije	191
Koliko je u relanomrealnom vremenu trajao runtime?	191
A sad ispričajmo u kratko što su radili naši objekti	192
Još jedan pogled na cijeli naš program - index.php	193
Ostale vježbe	193
Što slijedi	194
N a s t a v n a c j e l i n a 9 CMS	195
Metodika-----	195
Udžbenik-----	195
Što ćemo naučiti?	195
Što je CMS	196
Za one koji ovdje počinju raditi po knjizi	197
CMS – instalacija	197
CMS - administriranje	197
Upload slika	200
Promidžba	201
Programirajmo CMS	202
Vježba 1	202
Baza podataka	202
prazna.sql	202
Nakon što smo unijeli podatke naša baza izgleda ovako	203
Relacije među tablicama	205
Vježba 2	207
Rješenje	207
Objašnjenje	211
Vježba 3	215
Objašnjenje	217
Vježba 4	218
Kreiranje URL	218
index.php	218
Linkovi	219
Vježba 5	222
Objašnjenje	224
Vježba 6	227
Vježba 7	228

Vježba 8	234
Tražilica	234
Vježba 9	238
Objašnjenje tražilice	238
Zadatak	239
Zadatak	239
Vježba 10	239
SQL Join	239
Vježba 11	240
Administracija CMS	240
templates	242
smarty_plugins	242
Sažetak	243
N a s t a v n a c j e l i n a 1 0	E-trgovina 245
Metodika-----	245
Udžbenik-----	245
Što ćemo naučiti?	245
Baza podataka	245
trgovina.sql	245
PHP Smarty PEAR kôd	246
Plaćanje	246
Promjene koje je potrebno učiniti u postojećem kôdu cms-a	247
index.tpl	247
Dodaj u košaricu	248
jedinica.tpl	248
jedinica_kratko.tpl	248
Administracija e-trgovine	248
E-trgovina u radu	249
Sažetak	250
Literatura	250



Od Aristotela do Boolea i Gatesa

O čemu je riječ

1. Povijest softwarea po Ivici Kartelo
 - a. Aristotel i Boole
2. Povijest hardwarea po Ivici Kartelo
 - a. Prekidač
3. Iskustvo autora
4. Moji razlozi za ovakav tekst

Povijest softwarea po Ivici Kartelo

Aristotel

300. g. prije Krista, Aristotel je podučavao Aleksandra Velikog. U slobodno vrijeme je razmišljao i pisao. Između ostalog i Logiku. Potrebne su najmanje DVIJE premise za zaključak. (Računalstvu su potrebne samo DVIJE vrijednosti: 1 i 0). Primjer:

1. Ljudi se od kiše štite kišnim ogrtačem.
2. Danas je padala kiša.

Danas su ljudi nosili kišne ogrtače.

Boole

Dva milenijuma kasnije, u zabiti Engleske, učitelj Boole čita Logiku. Dolazi na ideju prevesti Logiku na jezik matematičara. Opredjeljuje se za binarni matematički sustav. S jedinicama i nulama pokušava objasniti ono što je Aristotel riječima. Tako nastaje Booleova algebra. Nastaje kraj lojanice, puno prije nego je izmišljena električna struja.

Primjer 1:

1. Kad voda curi kroz cijev to je 1.
2. Kad zatvorimo cijev to je 0.

Prekidač, zatvarač, brana može imati DVA stanja: otvoreno (1) ili zatvoreno (0).

Primjer 2:

1. U životu ima stvari koje moramo učiniti (I).
2. U životu ima stvari koje možemo birati (ILI).

Ako želimo dobiti građevinsku dozvolu moramo priložiti potvrdu jedan I potvrdu dva I potvrdu tri I potvrdu četiri I

Ako želimo doći do škole, možemo ići ovim putem ILI onim putem ILI ...

Tako su nastale tablice istine I i ILI.

I				ILI		
IZVOR--->A----->B---->C				IZVOR-----A----->C IZVOR-----B----->C		
A	B	C		A	B	C
0	0	0		0	0	0
0	1	0		0	1	1
1	0	0		1	0	1
1	1	1		1	1	1

Povijest hardwarea po Ivici Kartelo

Gates

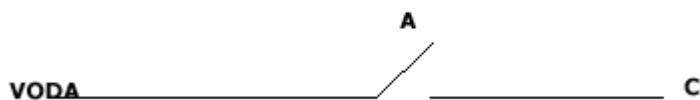
Vrata. Vrata su hardware koji je potreban za Booleovu algebru. Vrata imaju dva stanja: otvoreno (1) i zatvoreno (0).

U hidraulici i pneumatici se ta vrata još zovu ventili. U sustavu navodnjavanja su to brane.

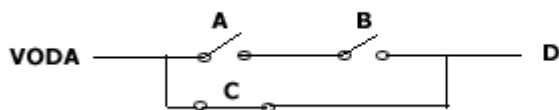
U elektrotehnici su to prekidači. Imamo ih kraj vrata i pomoću njih palimo i gasimo rasvjetu.

Zadaci

1. Hoće li u primjeru na slici teći voda u C?



2. Hoće li u primjeru na slici teći voda u D?



3. Napiši tablicu istine za zadatak 1.
4. Napiši tablicu istine za zadatak 2.
5. Kako bi pomoću prekidača kreirao program "Građevinska dozvola" ako je za dobiti istu potrebno 10 potvrda?

Test

1. Iz jednog jezera se voda slijeva u drugo preko 10 slapova. Da li tih 10 slapova čine logički sustav ILI, ili logički sustav I?

2. Velika jezera u Kanadi su plovna za brodove samo ako su sva vodna dizala ispravna i u funkciji. Koji logički sustav (ILI, ili I) čine vodna dizala?

Iskustva autora

Ovo poglavlje mi je najzanimljivije poglavlje u radu s učenicima. I njima. Zadržao sam se nekoliko sedmica s prvim razredima srednje strukovne škole. Satovi informatike. Čekali smo računala. U razredima je bilo 30 i više učenika.

Primjeri su bili s vodom ili električnom strujom i žaruljom. Svi bi učenici sudjelovali tako što bi izlazili redom na ploču i ubacili jedan ili više prekidača kao zadatak sljedećem učeniku. Tako bi postajalo sve teže. Na kraju sata smo na ploči imali pedesetak prekidača i pravi labirint.

Anegdota 1

Radili smo na primjeru el. žarulje. Da li svijetli ili ne svijetli? Martin je povukao dva prekidača. Pozvao je Ivana da riješi zadatak, ali me je prije toga zamolio: "Profesore, Ivan se boji struje, mogu li ja njemu zadati s vodom".

Prekidač

Na fakultetu smo koristili izraz 'vrata' (gates). I ja sam, kao i Bill Gates svoje prve programe radio na papiru. Diplomirao sam 1974. godine, a na FESB je stiglo prvo računalo godinu dana ranije. Sad je izloženo na hodniku. Samo su rijetki sretnici iz moje generacije dobili diplomski na njemu.

Vrata su oduvijek bila i ostala osnovni element računala. Što je stanica u ljudskom tijelu, to je prekidač u računalu.

Na kraju sata bih pokazao rukom na ploču punu prekidača i žarulja i rekao: "To je računalo! I što je najzanimljivije, napravili ste ga vi, ne ja!".

Memorija

Memorija je također prekidač. Pritisnite prekidač kraj vrata i on će ostati u tom položaju sve dok ga ponovo ne pritisnete. On ostaje u zadanoj poziciji. Jedna pozicija je 0, a druga 1. To znači da on čuva, pohranjuje, memorira 1 ili 0. Ako škola ima 1000 prekidača, njihovo stanje je ovog trenutka možda ovakovo:

00111111000000011101101101010101...

Priznat ćete da je to jedna super memorija. Mi ne bismo mogli zapamtiti niti prvih desetak pozicija prekidača, a sustav prekidača škole memorira 1000 pozicija!

Tehnologija

Povijest hardwarea je povijest prekidača. Prvi prekidači su bili ljudskom rukom pogonjeni, pa su i prva računala bila na ruke, ručna.

Zatim su prekidači postali elektromehanički, pa vakumske cijevi i na kraju poluvodiči, što su i danas.

Tranzistor

Prvi prekidač napravljen od poluvodiča je nazvan tranzistor. Tranzistor je zrna pijeska, silicija, u kojem se ništa ne miče i ništa se ne troši, pa živi vječno! To je bila revolucija.

Od te 1953. godine kad je tranzistor izmišljen, pa do danas, događa se samo evolucija u tehnologiji, koja se može definirati jednom riječju: minijaturizacija. Prvi tranzistor je bio velik kao palac, a danas je toliko velik mikroprocesor.

Procesor

Mikroprocesor je komad poluvodiča u kojem se nalazi nekoliko desetaka milijuna tranzistora.

Danas u firmi Intel sjedi nekoliko tisuća mojih kolega, inženjera elektrotehnike, svaki nagnut nad svojom pločom prekidača. Svaki od njih crta prekidače za koje je zadužen, baš kao što smo ih i mi crtali na ploči. A onda ih skupe sve na hrpu, sprže u jednom komadiću kamena i to je nova verzija procesora.

Moji razlozi za ovakav Uvod

Cilj

Što je bio moj cilj kad sam birao gornje priče i riječi? Cilj mi je bio:

- Osloboditi učenike predrasuda o računalu kao nečem pametnom i nedostupnom za svakog učenika.
- Sjećam se moje generacije u gimnaziji. Svi smo govorili da ćemo biti psiholozi, filozofi. Tehnika je bilo nešto pre šturo, ne kreativno, dosadno za nas. Srećom smo na vrijeme uvidjeli svoju grešku. Uključivanjem Aristotela u računalstvo vidimo da je tehnika jako povezana s filozofijom i logikom. I to jednostavnom ljudskom logikom.
- Michelangelo i Leonardo Da Vinci bili su inženjeri koliko i umjetnici. Izvrsni matematičari.

Aristotela sam uveo da pokažem kako je računalu od pamtivijeka dio ljudske logike. Logika. Naša svakodnevna logika. Kad pada kiša ponijet ću kišobran. Kad to mi kažemo nikom ništa, ali kad to ispiše računalu na ekranu ono je jako pametno. Toliko je pametno da me je strah i prići mu a kamoli ga učiti!

Međutim, u mojoj priči Aristotel je djed kao i svaki drugi, prost i narodni, uz lojanicu i stado ovaca i zamislite on je izmislio računalu. Kad može moj djed mogu i ja.

Hardware. Podignite poklopac i imate što vidjeti. Pa tko bi to mogao razumjeti! To nije za ljudsku glavu. A prekidač kraj vrata je za ljudsku glavu. Koja kamuflaža! Nešto nevjerovatno! Ali da je svih tisuću prekidača vaše škole staviti u jednu poveću kutiju, zajedno s njihovih 2000 žica, pitao bih vas ja onda bi li razlikovali računalu od te kutije! Za koju kutiju biste rekli da je pametnija?!

Svaka, ama baš svaka 'kutijica' u računalu je skupina minijturnih tranzistora. Kutijice imaju bakrene nogice s kojima se drže za bakrene stazice. Najpoznatije kutijice u računalu su RAM i PROCESOR. Ostalo su pomoćno osoblje.

Moji razlozi za ovu knjigu

Sve moje knjige su bazirane na primjerima, pa tako i ova. Nikad s nijednom knjigom nisam imao ambicije dati sve moguće iz date teme. Ni govora. Moje su knjige do 200 stranica. Sve preko toga su bile iznimke. U sebi sam priželjkivao pisati knjige koje će sadržavati sve, i vježbe i referentne priručnike, ali su prioriteti i mogućnosti diktirale što ću i u kolikom obimu pisati.

Danas, nakon toliko godina Interneta, navikli smo se na nj. Pripremamo se pomoću Interneta, točnije Googlea, i knjiga s Amazon.com. Na www.php.net ima sve o PHP. Na www.mysql.org ima sve o MySQL. Ako tome dodate relevantne liste Googlea na konkretne vaše upite, stvarno ljudi pa što ja to imam pisati u svojoj knjizi.

I te kako imam. Moje iskustvo, moje misli nećete naći na Internetu. Moje razmišljanje o podučavanju programiranja. Zato, za sve dijelove koji vam budu internetski, oprostite. To znači, moja vas knjiga još više gura prema Internetu i ostalim knjigama. To mi je bio jedan cilj.

Dalje, zanimalo me fenomen otpora prema programiranju, te dosadašnja naša praksa podučavanja programiranja.

Programiranje je mlada disciplina u školama. U svijetu i kod nas. Jer, PC se 'rodio' 1981., a Web 1990. PHP 1995. U školama su se učili samo elementi programiranja:

1. Instrukcije.
2. Petlje.
3. IF

Gotovih aplikacija koje se vide na tržištu, u školama se nije moglo vidjeti. U mojoj knjizi su upravo aplikacije kao finalni proizvodi u centru: Blog, Knjiga gostiju, CMS, E-trgovina.

Preporučujem polazak baš od tih aplikacija u radu. Neka ih učenici vide, isprobaju prije nego su tipkali svoju prvu instrukciju. Možemo reći, to je ono što ćete napraviti. Da bi to znali, trebate naučiti ove tehnologije:

1. XHTML
2. CSS
3. PHP
4. MySQL

U ovoj knjizi je obrađeno PHP i MySQL, na razini mog iskustva. Sve ostalo pitajte Googlea.

Mene zanima kako razmišlja čovjek koji programira. Došao sam do zaključka kako u centar treba staviti VARIABLE. Što je to u stvari? Mi radimo i stalno nešto pohranjujemo, spremamo, po registratorima, sve više u računalu, po ladicama. To uzimamo kad nam treba, mijenjamo, vraćamo u ladicu. Sve što spremimo obilježimo, imenujemo.

To radi i PROGRAM. Program RADI i POHRANJUJE, uzima, vraća, mijenja. FUNKCIJE programa RADE. Onome što pohranjuje on dodjeljuje imena i ta imena su VARIABLE. Pod imenom varijable pohranjuje tekst, brojeve, tablice, slike, glazbu itd. u RAM. Poziva te varijable. Mijenja im vrijednosti. Briše ih. Sprema ih u baze. Donosi vrijednosti iz baze i pohranjuje u RAM pod imenom svoje varijable. I o tome misli programer. Ako želi zbrojiti X i Y, programer mora odlučiti i tko će i kojim putem stavljati brojeve u ladice RAM-a X i Y.

Računalo na ploči kreacija učenika

Svaki učenik koji je izišao na ploču znao je riješiti zadatak koji mu je zadao prijatelj. Pratio bi prstom crte od izvora do žarulje i ako prolaz postoji, učenik bi označio žarulju da svijetli, u suprotnom bi obrisao znak da žarulja svijetli. Svaki učenik je riješavao dva tri prekidača, ovisno o inspiraciji njegova prethodnika. Međutim, tko bi na kraju sata ušao u razred i ugledao 50 prekidača i desetak žarulja i izvora, dobro bi se uznojio dok bi riješio koja žarulja svijetli a koja ne. Vjerojatnije je da bi se uplašio i pobijegao!

Ista je stvar s procesorom. To su milijuni prekidača odjednom ispred naših očiju. Ali oni koji su to napravili radili su jedan po jedan!

Pouka

Neke ću stvari namjerno ponavljati kroz knjigu samo da ukažem koliko su važne, iako na prvi pogled banalne i svima poznate. Praksa 'korak po korak' (prekidač po prekidač) neka vas prati uvijek kad programirate.

Što je PREKIDAČ u hardwareu, to je INSTRUKCIJA u programiranju. To su najmanje jedinice, prva hardwarea, druga softwarea. KAD NAPIŠETE JEDNU INSTRUKCIJU ISPROBAJTE PROGRAM KAKO RADI PRIJE NEGO NAPIŠETE DRUGU INSTRUKCIJU!

Ako se pojavi greška znate gdje je! U zadnjoj napisanoj instrukciji! A onda je puno lakše baviti se tom jednom instrukcijom, pregledati sve znakove, razmaknice, navodne znakove itd., nego s njih desetak ili stotinu.

Učenje na daljinu mi se tu pokazalo puno učinkovitije od razredne nastave. Ovo ljeto sam mom 15-godišnjem učeniku uvijek ponavljao jedno te isto:

Budi strpljiv. Obriši zadnju instrukciju koju si napisao pa isprobaj program u radu. I sve tako dok ne dođeš do zdravog dijela programa. E sad piši ponovo instrukciju, pa probaj u radu, piši instrukciju, pa probaj u radu i sve tako do kraja. Ako se greška pojavi znat ćeš u kojoj je instrukciji i to je to!

I sad te ja pitam, što je u učenju programiranja važnije: naše strpljenje ili naša pamet?! Pameti svi imate, ali će naučiti samo oni od vas koji aktiviraju i svoje strpljenje.

To treba isticati. Jer, učenicima je najlakše proglasiti programiranje baukom i teškim, još bolje, dosadnim, što treba eksplicitno zaustaviti. Ivane, kad budeš hladnokrvno i smireno sjeo za računalo, kad se naoružaš strpljenjem, bit ćeš genijalni programer, kojem će se softwareske firme jagmiti dati posao!

Podučavatelji, imajte svoje osobne web stranice

Zato se od svakog podučavatelja danas očekuje da ima i svoju osobnu web aplikaciju na kojoj će objavljivati teoriju, zadatke, testove da mogu učenici neke stvari odraditi mirno i strpljivo kod svoje kuće (na daljinu). Uz ovu knjigu imate i CMS web aplikaciju koju možete koristiti u tu svrhu. Uvijek se možete obratiti i autoru knjige za pomoć, na email info@e92.hr, koji još nikad nije propustio priliku odgovoriti svojim kolegama, kao i svim ostalim svojim čitateljima.

Sažetak

Programiranje je posao kao i svaki drugi. Za učiti programirati treba nam strpljenje. Izumitelji programiranja Aristotel i Boole, živjeli su puno prije nego je izmišljena električna struja. Za njih je programiranje bila humanistička, ne tehnička aktivnost. Bilo je filozofija. Za nas danas, programiranje je dobar, zanimljiv i kreativan posao, dobra plaća. A kad nešto znamo, nije nam teško. Volimo raditi ono što znamo.



PHP za početnike

Metodika -----

Windows je najrašireniji sustav u školama i u kućama. Zato je najjednostavnije skinuti s Interneta paket XAMPP za Windows i instalirati Apache, MySQL i PHP. U Google napišite XAMPP i doći ćete do adrese. Instalacija na Windows XP traje nekoliko minuta. Na sva pitanja odgovorite potvrdno. Tako se svako računalo u razredu nađe u okruženju otvorenog kôda i učenje programiranja može početi. Isto će učiniti učenici na svojim računalima kod kuće i tako stvoriti uvjete za izradu domaćeg rada, učenje na daljinu, vježbanje.

Mogući problemi prilikom instalacije PHP

Imao sam problem. Svi mogući problemi su odlično obrađeni na Internetu i naći ćete ih pomoću Google. Riješenje mog problema je bilo u ponovnoj instalaciji Windowsa XP. Nakon toga je instalacija protekla uobičajeno.

Prva aplikacija

```
<?php
phpinfo();
?>
```

Koristim uobičajeni početni kôd PHP-a. Pokazalo se vrlo korisnim što svi u svijetu koristimo isti početni kôd a datoteci svi dajemo ime test.php. Ako vam ne radi taj prvi kôd, idete u Google i napišete test.php i evo vam svih mogućih situacija. Možda je i jedna od njih vaša.

Objašnjavamo sintaksu kôda.

Instrukcija

To je osnovna jedinica u svakom programu. Što je PREKIDAČ (tranzistor) u hardwareu, to je INSTRUKCIJA u softwareu.

Svaka instrukcija ima točku zarez na kraju; U jednom retku se može pisati više instrukcija. Neki retci nemaju točku zarez, kao na primjer početak IF strukture, petlje itd. što ćete i vježbati kroz primjere.

Instrukcije su dakle konvertirane u binarne zapise:

```
111010101010101111111010100111011
```

```
000011011110001001100100101010101
```

itd.

i procesuirane jedna po jedna te uništavane za svagda. Sljedeći zahtijev za stranicom i sve se ponavlja: kompajliranje, izvođenje, uništavanje.

HTML (XHTML) i CSS

U Web pregledniku dobivamao impresivni HTML dokument. U ovoj knjizi ima puno HTML, preciznije, XHTML-a, ali samo u kôdu primjera. Druge knjige obrađuju XHTML. Isto vrijedi i za CSS.

Druga PHP aplikacija

Ovdje sam napisao cijeli XHTML dokument i tako istakao poziciju PHP kôda u tom dokumentu. PHP je uvijek između ovih graničnika `<?php ?>`. Kad Apache dobije zahtijev za web stranicom koja ima nastavak `.php`, ne isporučuje ju takvu kakva je, nego ju prosljeđuje PHP interpreteru. PHP interpreter preskače XHTML i počne raditi tek kad naiđe na prvi graničnik `<?php`.

Tad počinje interpretiranje. Tad počinje kompajliranje. Tad počinje runtime programa. Tad počinje procesuiranje programa. Što to znači? Jedan PHP program je sve što se nalazi između graničnika `<?php ... ?>`. PHP program je skriptni program što znači da je čitljiv i vidljiv svakome tko dođe do originalnog `.php` dokumenta, web stranice na serveru.

Interpreter kompajlira taj skriptni jezik. Što znači 'kompajlira'? To znači da ga prevede na jezik kojeg razumije računalo, a to je binarni kôd:

```
1110001010101001010101
```

```
0100100101001010010100
```

```
1010010101010100000101
```

itd.

Instrukcije u binarnom kôdu dospiju u RAM i program je postao realnost, događa se, živi, izvodi se. Procesor uzima instrukciju po instrukciju iz RAM-a i obrađuje ih. Rezultate obrade pohranjuje u RAM. To su razne varijable i konstante. Sljedeća instrukcija može koristiti varijable iz RAM-a koje je ostavila prethodna instrukcija i sve tako. Svaka procesuirana instrukcija biva definitivno uništena. I tako procesor uništava redom sve instrukcije. Kad dođe do graničnika `?>` zna da je to kraj programa i pomete sve tragove o tom programu u RAM-u. Program je mrtav. Runtime je završio. Program je živio od prve do zadnje instrukcije u RAM-u.

Rezultat PHP programa je XHTML koji se dodaje već postojećem XHTML-u i sve zajedno je jedan XHTML kojeg Apache vrati pregledniku i prekine svaku vezu s preglednikom.

Ističem!

```
<?php
phpinfo();
?>
```

Ovdje se ipak moramo malo zadržati. Ovo je knjiga o programiranju u programskom jeziku PHP. PHP se piše unutar web stranice, preciznije, unutar XHTML-a i CSS-a. U primjerima koji sljede to će biti puno očitije.

Na primjer:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
Ovo je prvi PHP program:
<?php
echo "Prvi!";
?>
Ovo je drugi PHP program:
<?php
echo "Drugi!";
?>
Ovo je treći PHP program:
<?php
echo "Treći!";
?>
</body>
</html>
```

Istakao sam tri PHP programa, kao tri 'otoka' u 'moru'. Svaki će biti procesuiran. Ali su sva tri na JEDNOM fizičkom dokumentu. Taj jedan fizički dokument je u cijelosti PREDAN u 'ruke' INTERPRETERU. Zato, s krajem PRVOG php programa NIJE kraj i interpretacije fizičkog dokumenta. Zato, znakovi ?> ne znače i kraj SVEGA što se događalo u php programu. Varijabla rođena u jednom otoku ŽIVI do kraja STRANICE, a ne do krajnjeg graničnika ?>.

Neka ta stranica ima datotečno ime index.php. Kad pozovemo stranicu koja ima nastavak .php, na primjer index.php, Apache predaje takvu stranicu PHP interpreteru. Cijela ta WEB stranica je JEDAN PROGRAM. Ona cijela prolazi kroz INTERPRETER, ali se interpretiraju samo instrukcije između znakova <?php ... ?>. Sve izvan tih znakova ostavlja se kako je. Dijelovi izvan tih znakova su finalni dijelovi buduće WEB STRANICE. Ono što nazivamo GLOBALNOM varijablom to 'živi' koliko i cijela web stranica. Globalna varijabla kreirana u PRVOM programu ostaje u RAM-u i nakon što je taj PRVI PROGRAM interpretiran, procesuiran. Tek kad se prođe posljednji tag </html> web stranice, PRESTAJE runtime te stranice i interpreter BRIŠE u RAM-u sve varijable nastale u pojedinim programima na toj stranici.

Kasnije u ovoj knjizi, na poglavlju CMS, vidjet ćemo kako se PHP kôd izdvaja od XHTML. U jedan fizički php dokument pišemo jedan php program radi preglednosti i boljeg snalaženja.

Zato je moguće da ću u knjizi ponekad reći i ovako: runtime počinje `<?php ...` a završava ovdje `?>`. Nakon znaka `?>` nema više ništa u RAM-u od tog php programa. To vrijedi SAMO kad je JEDAN php program na jednom fizičkom dokumentu.

Što istaknuti u ovoj početnoj fazi

Što istaknuti u ovoj fazi učenja programiranja pokazao sam u zadacima i testovima. Za ovu knjigu je svakako dobro došlo predznanje XHTML i CSS. Ali i bez toga, početnik u programiranju se može koncentrirati samo na ono što se događa između graničnika `<?php ... ?>`.

Dok su programi još 'mali otoci' u XHTML-u dobro bi bilo govoriti o cjelovitom procesu od trenutka kad preglednik pozove web adresu do trenutka kad mu Apache vrati XHTML dokument. Kad web stranica ima nastavak `.htm` ili `.html`, Apache je odmah vrati takvu kakva je. Kad dokument ima nastavak `.php`, događa se gore opisano. Kad radimo web aplikacije koje uključuju više php stranica, svima stavljamo `.php` nastavak pa i kad to ne bismo trebali jer ima stranica koje ne sadrže PHP na sebi, nego samo XHTML. To je dobra praksa jer uvijek naknadno možemo u takve stranice dodati PHP kôd pa da se ne opterećujemo s različitim nastavcima i o tome još vodimo brigu. To neće niti malo vidljivo opteretiti brzinu učitavanja.

Posebno je važno ISTAKNUTI i staviti na svoje mjesto navedene pojmove:

- interpretirati
- binarno
- kompajlirati
- runtime
- život programa
- procesuiranje programa
- izvođenje programa
- otvorite izvorni kôd pristigle web stranice i uvjerite se kako nema PHP kôda.

Varijable

Ne treba puno govoriti o tipovima varijabli, nego ići na primjere i kroz primjere objašnjavati i značaj varijabli. Jedna od 'dosadnih' i 'tlaka' tema je 'ma zašto trebam razbijati glavu o tipovima varijabli?

Tu treba jedno razjasniti, a to je: učiteljica je u prvom osnovne najviše vremena utrošila za naučiti nas razlikovati brojeve od slova. To smo zaboravili. Pa zašto bi računalni program u svom kratkom životu od znaka `<?php` do znaka `?>` trebao još i gubiti dragocijeno vrijeme na učenje razlike između brojeva i slova, slika i glazbe itd.

Zato mi njemu u tome pomognemo kad god možemo i nema zabune. Primjeri na tu temu tek sljede.

Za početak je važno koncentrirati se na samu varijablu \$x i ne pomagati PHP-u. U jednostavnim primjerima on razlikuje brojeve od slova, pa smo počeli od takvih primjera. Kad stavimo navodnike oko nečega, to je za PHP string (tekst). Oko brojeva ne moramo staviti navodnike, a oko imena varijabli ne smijemo. Ako oko broja i stavimo navodnike, PHP će ga ipak prepoznati kao broj tamo gdje treba. Ako stavimo navodnike oko varijable, PHP to neće prepoznati kao varijablu.

Lokalne i globalne varijable

Pitao sam se što je ključno za programiranje. Na koji način programer treba razmišljati? Kako najučinkovitije prijeći prag od neznanja, straha, omraženosti do znanja i kreativnosti?

Prema dosadašnjem iskustvu, ističem ovo. Programiranje je preslika našeg svakodnevnog života. Mi obavljamo neke funkcije, spremamo slike, dokumente po ladicama, pa ih vadimo i opet spremamo. Ne samo slike i dokumente, nego i robu, posude, namještaj. U društvu spremamo svoje riječi u tuđe i svoje glave. Ja sad preko ove knjige spremam ovaj tekst u vašu glavu. Tražimo onaj dokument... nema ga, ma gdje smo ga spremili? U ovom što sam rekao su dvije ključne riječi: SPREMIO i FUNKCIJE (činiti nešto).

Te dvije ključne riječi, SPREMATI i UČINITI su nam na pameti od prvog retka programa do zadnjeg. Što su nama ladice, ormari, sobe ili disk našeg računala, to je programu VARIJABLA. Program pohranjuje u VARIJABLU. Fizički je ta varijabla uvijek u RAM-u i o tome ne treba misliti. Treba misliti o VARIJABLAMA. Program je PRIČA, POSAO kojeg treba obaviti. A tko su likovi u toj priči, izvođači, glumci? To su varijable. I o tome mi mislimo kad mislimo o programu. Varijable su ladice u RAM-u.

Želimo zbrojiti vrijednosti u ladicama X i Y i rezultat upisati u web stranicu, pišemo ovaj program:

```
<?php
echo $x + $y;
?>
```

Program će izbaciti rezultat '0'. To nam je jasno jer nismo kreirali ladice u RAM-u koje se zovu \$x i \$y. Nema ni ladica, pa nema ni vrijednosti.

```
<?php
$x
$y
echo $x + $y;
?>
```

Sad nam program izbacuje prazan list. Nema niti nule. Imenovali smo ladice u RAM-u, ali u njih nismo stavili nikakvu vrijednost. Očito je ovo POGREŠKA na koju PHP reagira prekidom rada.

```
<?php
$x;
```



```
$y;
echo $x + $y;
?>
```

Sad je izbacio opet nulu 0. Znači, možemo mi kreirati 'ladice' u RAM-u, prazne, ali se INSTRUKCIJA mora završiti s točkom zarez ;. Nismo stavili točku zarez iza \$x i \$y i to je bila greška na koju je PHP reagirao prekidom rada.

```
<?php
$x = 1;
$y = 2;
echo $x + $y;
?>
```

Rezultat je 3. Oko 1 i 2 nismo stavili navodnike. Ako i stavimo, PHP će opet izbaciti 3.

```
<?php
$x = "1";
$y = "2";
echo $x + $y;
?>
```

Postupak procesuiranja teče ovako:

- echo moram ispisati na web stranicu ono što sljedi
- \$x idem u RAM, u ladicu na kojoj piše \$x, otvaram je, čitam broj koji vidim u toj ladici, to je broj 1, pamtim to i nastavljam dalje
- + ... znači ovu jedinicu (1) moram zbrojiti s onim što sljedi, da vidimo što je to
- \$y ... idem u RAM, u ladicu na kojoj piše \$y, otvaram je, čitam broj koji vidim u toj ladici, to je broj 2, pamtim to i nastavljam dalje
- ; ... gotovo, nema više, gotov sam s ovom instrukcijom koja glasi "napiši na web stranicu 1 + 2" i ja, procesor, stavljam u RAM broj 3 kao dio već postojećeg XHTML-a koji u RAM-u čeka na isporuku prema pregledniku koji ga je pozvao.

Lokalne varijable su sve varijable unutar funkcije. **Globalne varijable** su sve ostale. U gornjem primjeru, varijable \$x i \$y su GLOBALNE

Sad se postavlja pitanje tko sve može stavljati VRIJEDNOSTI u varijable. POHRANJIVATI vrijednosti u VARIJABLE.

U gornjem primjeru smo to učinili mi RUČNO. U praksi to čine FUNKCIJE. Na primjer:

```
$x = mysql_query("SELECT * FROM dobavljac");
```

Funkcija mysql_query() je pomoću SQL upita SELECT * FROM dobavljac vratila varijabli \$x cijelu jednu tablicu 'dobavljac', ma koliko velika ta tablica bila.

Na web stranici postoji FORMA u koju smo tipkali svoje ime i prezime. Evo kako naše ime i prezime može dospjeti u ladice \$x i \$y.

```
$x = $_POST["x"];
```

```
$y = $_POST["y"];
```

Ovdje smo mi opet RUČNO zadali vrijednosti varijablama \$x i \$y, ali ovaj put preko web stranice. To može učiniti svatko tko pristupi toj web stranici.

Forma

Svaki program može komunicirati s drugim programima i s ljudima. Za komunikaciju s drugim programima ne treba mu ono što mu treba za komunikaciju s ljudima. Što je to? To je sučelje za ljude. Danas je uobičajeno grafičko sučelje: za unos teksta, slanje datoteka, gumbi za 'Pošalji' i slično. Kad je riječ o web aplikacijama, takva sučelja se rade u XHTML i CSS tehnologiji. To je web stranica. Dijelovi web stranice koji služe za interakciju s aplikacijom nazivamo FORME.

Sve što upišemo u FORMU dospije u RAM našeg računala. Kad kliknemo na gumb Šalji, pokrenuli smo izvršavanje GET ili POST metode. GET ili POST metoda uzme taj naš unos iz RAM-a našeg računala i odnese to u RAM poslužitelja. Možemo li mi pohraniti bilo što na naše računalo a da tome ne dodijelimo ime? Ne možemo. Tako i programi. Sve što programi stavljaju i uzimaju iz RAM-a mora biti pod nekim imenom. To su VARIJABLE. Tako je naš unos iz forme dospio u varijablu \$_POST['x'], a drugi unos u varijablu \$_POST['y'], pa instrukcija glasi:

```
echo $_POST["x"] + $_POST["y"];
```

Mogli smo mi i ovako pisati program:

```
$a = $_POST["x"];
```

```
$b = $_POST["y"];
```

```
echo $a + $b;
```

Sve je to samo pitanje naše organizacije i koliko ćemo ladica RAM-a (varijabli) angažirati. Povirite u urede. Netko slaže u više mapa, netko u manje, a netko na hrpu na stolu. Bitno je da vlasnik zna gdje mu je što. Bitno je da naš program zna gdje mu je što.

IF nas je sve prevario

Nekoliko godina sam na web siteu držao sljedeći kviz:

Računalo je pametno kao:

- kamen
- traktor
- biljka
- životinja
- čovjek

80% svih odgovora glasilo je: čovjek. Tko nas je prevario, da tako pogrešno mislimo o računalima.

Prevarila nas je STRUKTURA IF. Niti jedan drugi dio programa, nego taj IF! IF je pametnjaković! IF reagira na vanjske i unutrašnje impulse. Pomoću IF računalo ima sva moguća osjetila:

1. spojimo na računalo senzor za vlagu i pomoću IF:

```
IF (isset($vlaga)
echo "Pada kiša";
else
echo "Ne pada kiša";
naše računalo vidi, osjeća vlagu izvan našeg stana!
```

Zato je IF jedna od najzahvalnijih tema u programiranju. Kad IF povežemo s LOGIČKIM I USPOREDNIM OPERATORIMA dobijemo sigurno jedno od najuzbudljivijih tema u programiranju nakon vježbi na ploči s onim prekidačima i žaruljama koje svijetle ili ne svijetle.

Sljedeće tako veliko i zahvalno i uzbudljivo područje u programiranju nam dolazi tek s objektno orijentiranim programiranjem.

Ovdje treba dobro aktivirati maštu učenika, kao i svoju učiteljsku maštu, i izmišljati i riješavati što više zadataka. Na IF strukturama leže sve aplikacije ovoga svijeta bez obzira na jezik i autora.

Ističem kôd IF

```
if ($_POST["x"] < 20)
{
print "Vi imate manje od 20 godina.";
}
else
{
print "Vi nemate manje od 20 godina.";
}
```

Ovaj redak `if ($_POST["x"] < 20)` uvijek znači "Ako je to u zagradi ISTINA" ili "Ako je to u zagradi jednako 1" ili "Ako je to u zagradi jednako TRUE".

I sad je lako ponoviti sve što smo radili s PREKIDAČIMA, tj. vježbati logičke I, ILI. Gleda se istinitost svake zgrade posebno, pa onda ukupna istina svih zgrada zajedno:

Primjer 1:

```
if (($_POST["x"] == 20) || ($_POST["y"] == 20))
{
print "Ono u zagradi je TRUE";
}
else
{
print "Ono u zagradi je FALSE";
}
```

Gleda se ovako:

if ((\$_POST["x"] == 20) || (\$_POST["y"] == 20)) = If (1 || 0) = if (1)

if ((\$_POST["x"] == 20) || (\$_POST["y"] == 20)) = If (0 || 0) = if (0)

if ((\$_POST["x"] == 20) || (\$_POST["y"] == 20)) = If (1 || 1) = if (1)

if ((\$_POST["x"] == 20) || (\$_POST["y"] == 20)) = If (0 || 1) = if (1)

Primjer 2:

```
if ((($_POST["x"] !== 20) || ($_POST["y"] == 20) && ($a == 4)))
{
    print "Ono u zagradi je TRUE";
}
else
{
    print "Ono u zagradi je FALSE";
}
```

Gleda se ovako:

if (((\$_POST["x"] !== 20) || (\$_POST["y"] == 20))&& (\$a == 4)) =

if ((1 || 0) && 1) = if(1&&1) = if(1)

=====

if (((\$_POST["x"] !== 20) || (\$_POST["y"] == 20))&& (\$a == 4)) =

if ((0 || 0) && 1) = if(0&&1) = if(0)

if (((\$_POST["x"] !== 20) || (\$_POST["y"] == 20))&& (\$a == 4)) =

if ((1 || 0) && 0) = if(1&&0) = if(0)

Izvrсна prilika za uvježbati TABLICE ISTINE AND i OR.

Petlje

Petlje su vrijedni radnici bez kojih bi se loše proveli. Petlje će kreirati po želji nizove od nule, ili će nam pronaći željeni član nekog jako velikog niza, u praksi je to najčešće neka jako velika tablica.

Što se vježbanja tiče, vrijedi isto što i za IF strukture. Ovdje se zaustaviti koliko treba da svi učenici dobro nauče koristi PETLJE!

Moja iskustva

Vježbe

Stalno i uvijek vježbe, vježbe. Pisanje kôda, pisanje kôda. Kako će pisati kôd ako nisu čuli i naučili teoriju? To ide zajedno.

Pripremao sam se pomoću Interneta. Našao bih odgovarajući kôd ili pak ideju, prilagodio ga svojim potrebama, mijenjao sasvim ili jako malo. Tiskao bih na papir predviđeni kôd za taj dan i kopirao za svakog učenika primjerak, koji je nerijetko imao i nekoliko listova A4.

Podijelio bih papire učenicima i svaki je učenik tipkao u računalo primjere. Isprobavao da li radi. Ako ne radi, tražio bi grešku. Od učenika kojem primjer radi, kopirali bi oni kojima ne radi i s njim usporedili svoj kôd i tako pronalazili grešku, udruženo.

Predviđene sekcije za teoretsko objašnjenje bih puštao na projektor ili ispisao na ploči ili jedno i drugo i objasnio. To se posebno odnosilo na prve nastavne cjeline i teme kao što su:

- Instrukcija.
- Graničnici `<?php` i `?>`.
- Varijable.
- Život varijabli.
- Program. Život programa (runtime).
- Varijable znače POHRANJIVANJE vrijednosti u RAM pod imenom varijable. Vrijednosti možemo dati mi ručno, ali programi rade tako da ne trebaju naš 'ručni' rad, već samostalno kreiraju varijable, mijenjuju pohranjene vrijednosti u njima i sve tako od početka do kraja rada programa. Jasno, sve smo to MI ISPROGRAMIRALI.

Ukoliko bi zadaci bili povećati, na nekoliko listova, učenici bi se podijelili u grupe i onda bi svaki učenik tipkao dio programa. Spojili bi to i ako ne radi tražili grešku USPOREDBOM s onim na papiru.

Dešavalo se da i ja pogriješim na papiru. Zato bi mjerodavan bio onaj rad učenika koji prvi ispravno radi i tad smo imali uzorak s kojim se možemo uspoređivati.

Ja sam na USB-u uvijek imao verziju koja radi, pa ako treba, dao bih učeniku moju verziju koja radi da usporedi sa svojom i pronađe grešku.

Na računalima bi ostajali ti radovi učenika, pa bi sljedeća grupa učenika odmah otkrila te radove i ne bi tipkala svoje nego uzela te gotove.

Zato sam pripremao različite kôdove za svaku grupu.

Kad učenik ima svoj papir, radi svojom brzinom, zadržava se gdje hoće, planira svoj tempo.

Kad gledaju i prepisuju s projekcije, brži čekaju najsporije da se projicira sljedeći pasus. Ne mogu samostalno tražiti greške.

Na ploču sam pisao samo kad je o tome trebalo diskutirati, ili kad bih nešto htio objasniti.

Te papire su spremali u mape i imali su po čemu vježbati i kod kuće. Na kraju svakog polugodišta, svaki učenik mi je trebao pokazati mapu sa svim papirima.

PDF

Pošto je to ipak veliki trošak za školu, kopirati toliko papira, rekao sam svim učenicima da kupe i nose sa sobom USB memorije. Počeo sam donositi na svom USB-u sve češće PDF format i kopirati ga na mrežu, tako da ga ima svaki učenik

na svom računalu. Tako je mogao prepisivati s PDF dokumenta, ali nije mogao kopirati (kao što bi mogao s Word dokumenta).

Moguće je također pripremiti kôd i u običnom PAINT-u i tako ga distribuirati. Važno je da učenik tipka kôd, a ne Copy/Paste.

Teorija (Diskusija)

Teoriju sam objašnjavao na primjeru, redak po redak, kratki primjeri, na način da učenici što više sudjeluju. Ustvari oni i slože cijelu priču tijeka programa, tako što odgovaraju na moja pitanja.

Primjer

Učenici su okrenuti prema ploči i računala su pogašena. Na ploči sam ispisao ovaj kôd:

```
<body>
<?php
$x = 1;
$y = 2;
echo $x + $y;
?>
</body>
```

Profesor: Idemo redom. Svatkom od vas ću postaviti po jedno pitanje i svi vaši odgovori zajedno bit će TEORIJA programiranja ovog konkretnog zadatka.

Ivane, možemo li reći da je PHP 'otok' u XHTML-u?

Ivan: Možemo, profesore.

Profesor: Koliko otoka imamo u ovom primjeru?

Vinko: Jedan otok.

Profesor: Koliko i koje su varijable u programu?

Marko: Varijable su x i y.

Profesor: Što znači ova instrukcija: \$x = 1;?

Ivana: To znači da je u RAM-u kreirana 'ladica' na kojoj piše \$x i u tu 'ladicu' stavljena je vrijednost 1. Isto vrijedi za varijablu \$y.

Profesor: Koliko dugo će živjeti te varijable?

Jelena: Te varijable će živjeti dok traje procesuiranje cijele web stranice. Bit će na raspolaganju svim php 'otocima' na toj stranici. Zato su to GLOBALNE varijable. Varijable unutar funkcija su LOKALNE.

Itd.

Usmeni

Dok učenici tipkaju kôd, prepisuju s papira ili kreativno rade svoj kôd, porazgovarao bih s pojedincima o kôdu. Razgovor bi imao za cilj da ja tom učeniku dodatno objasnim kôd, da saznam koliko zna. Mogu mu dati i papir s nekim

programom u kojem nedostaju neki dijelovi. Njegov je zadatak ispred mene popuniti te dijelove.

Primjer

Mogu mu dati papir s ovakvim pitanjem:

Što nedostaje ovom programu, ako želim da mi izbací broj 3:

```
<?php
$x = "1";
echo $x + $y;
?>
```

Dobrih odgovora uvijek ima više:

```
$x = 3;
$y = 2;
echo "1 + 2";
echo $_POST["a"] + $_POST["b"];
```

Imamo o čemu pričati čak i kod najmanjih mogućih programa. Meni osobno su najzanimljivije upravo ove priče o varijablama i tijeku informacije, fizički i na papiru.

Pismeni

Testovi bi bili drukčiji za svakog učenika. Rezultatski drukčiji. Radio sam probne testove i glavne testove. Rekao bih unaprijed datume kad je probni, a kad glavni test. Ocjena probnog ide u moj virtualni rokovnik, a glavnog u dnevnik. Pismeni su imali 20 zadataka, pitanja, pitanja s ponuđenim odgovorima i sl. Poneke zadatke su mogli riješiti na papiru, a neke su morali i isprobati na računalu, prije nego bi odgovorili.

Posebno su učinkoviti programi kojima ponešto nedostaje pa učenici trebaju dodati. Ili moraju ispisati cijeli program na osnovu opisnog zadatka.

Domaći rad

Učenici tijekom cijele školske godine rade osobne web aplikacije i timske po vlastitom izboru i timova i tema. O tome se stalno konzultiraju samnom.

Email

Svim učenicima je moj email na raspolaganju i mogu mi pisati kad žele.

Kao uostalom i vi dragi kolege i kolegice podučavatelji. Moj mail je info@e92.hr Još se nije dogodilo da nisam odgovorio na mail. Tako ova knjiga u sljedećem izdanju može biti samo bolja.

Udžbenik-----

Što ćemo naučiti u prvoj nastavnoj cjelini?

U prvoj nastavnoj cjelini ćemo instalirati PHP i Apache na svoje Windows računalo. Počet ćemo od jedne jedine programske instrukcije i naučiti prve ključne riječi print i echo. Slijede PHP varijable. Naučit ćemo koliko žive varijable, a koliko žive programi. Pomoću HTML forme ćemo upoznati najčešće korištene globalne varijable. Nastavnu jedinicu ćemo završiti s IF uvjetnom strukturom i petljama.

Instalacija PHP

Upozorenje: Ako već imate instaliran PHP i Apache preskočite ovo poglavlje. Kako ćete znati da li je na vašem računalu instaliran PHP i Apache? Ovako:

1. Start/Upravljačka ploča (Control Panel)/Administrativni alati (Administrative Tools)/Servisi (Service)/
2. Ako je na popisu servisa Apache(broj verzije) na vašem računalu je već instaliran web poslužitelj Apache. Najvjerojatnije je onda na vašem računalu već instaliran i PHP. Preskočite poglavlje o instalaciji i prijedite odmah na poglavlje 'Moje prve programske instrukcije'.

Zaustavite rad servisa IIS Admin

Provjerite postoji li na vašem računalu Microsoftov Web poslužitelj IIS. Ako postoji, zaustavite njegov rad ovako:

1. Start/Upravljačka ploča (Control Panel)/Administrativni alati (Administrative Tools)/Servisi (Service)/Klik na IIS Admin ako postoji u popisu servisa/Klik na Stop.

Tako ste pojednostavnili instalaciju Web poslužitelja Apache.

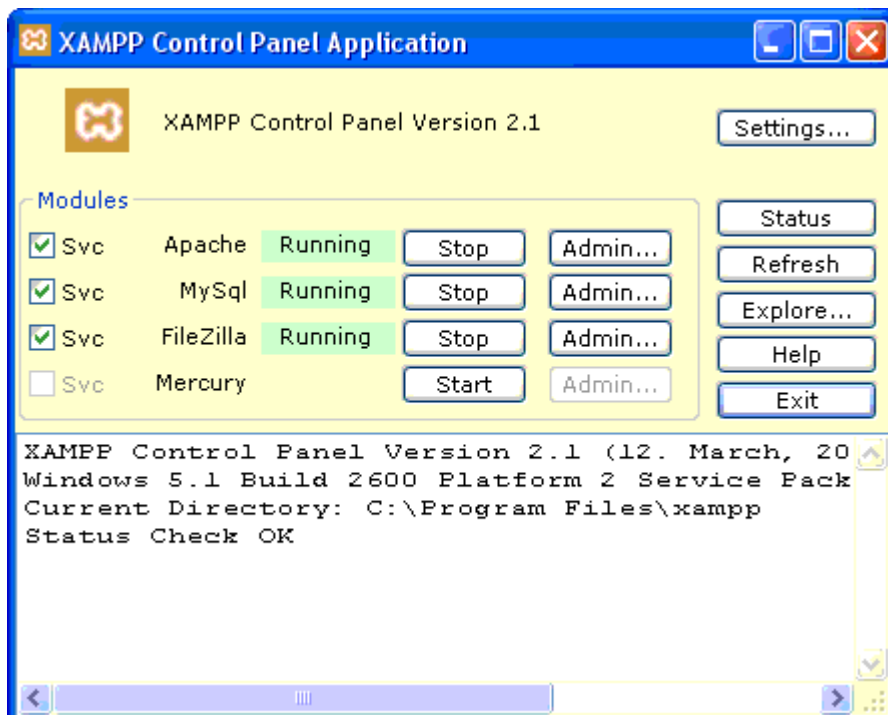
Instalacija Apachea i PHP

Apache je Web poslužitelj. PHP je programski jezik.

[Download](#) s Interneta XAMPP for Windows. To je najbolji Windows Installer Apachea, PHP-a i MySQL-a. Tipkajte u Google 'xampp' i dobit ćete url za download.

Kad je XAMPP jednom na vašem računalu, dva puta klik na isti, prihvatite i potvrdite sve što vam ponudi i za par minuta ćete imati sve potrebno za programiranje u PHP-u.

Ako na kraju instalacije na svom računalu vidite XAMPP Control Panel u kojem piše Apache Running, možete početi programirati u PHP.



Moje prve programske instrukcije u PHP-u

Prva PHP aplikacija

U Notepad napišite sljedeći PHP kôd:

```
<?php
phpinfo();
?>
```

Pohranite Notepad dokument pod imenom **test.php** u mapu **C:\Program Files\xampp\htdocs**

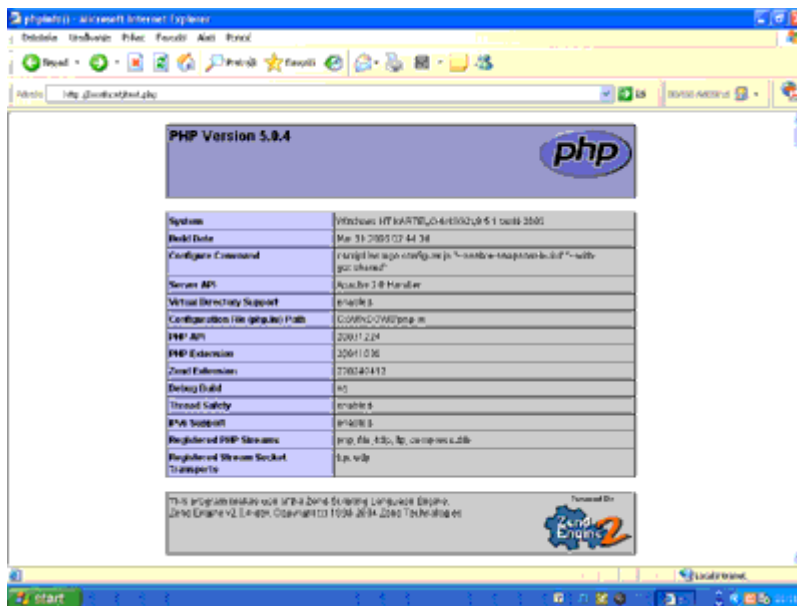
Napomena: dobra je praksa u polju File type ('Spremi u obliku') odabrati All files ili Svi dokumenti, jer se zna dogoditi da Notepad, suprotno našoj volji, doda još i nastavak .txt, pa će ime datoteke biti test.php.txt. Ukoliko je na vašem računalu podešeno da se nastavci ne vide, onda ćete u Windows Exploreru vidjeti samo test.php, a ne i onaj posljednji nastavak .txt. Da li ćete nastavak vidjeti ili ne, podešava se u Start/Control Panel/File Option.

Pogledajmo u radu našu prvu PHP aplikaciju

Otvorite Web preglednik.

U polje Address tipkajte ovu adresu: <http://localhost/test.php>

Enter



Objašnjenje moje prve PHP aplikacije

- PHP kodiranje uvijek počinjemo s `<?php` a završavamo s `?>`
- Kao svako programiranje tako i ovo s PHP se sastoji od instrukcija.
- Instrukcijama moramo na kraju pisati točku zarez ;

```
<?php
phpinfo();
?>
```
- U gornjem primjeru smo samo pozvali PHP ugrađenu funkciju `phpinfo()`.
- Ta funkcija se izvršila na Apacheu i Apache nam je poslao rezultat te funkcije.
- Rezultat te funkcije je impresivan HTML dokument s masu podataka o Apacheu, PHP-u i ostalom softwareu kojeg smo dobili s njima.
- Već na početku smo se uvjerali kako mogu biti moćne i učinkovite ugrađene funkcije.

Moja druga PHP aplikacija

U primjeru `test.php` rezultat funkcije je HTML dokument. Kad ne koristimo ugrađene funkcije koje će umjesto nas napisati HTML tagove, te HTML tagove moramo pisati mi:

1. U Notepad dokument tipkajte ovaj PHP programski kôd:

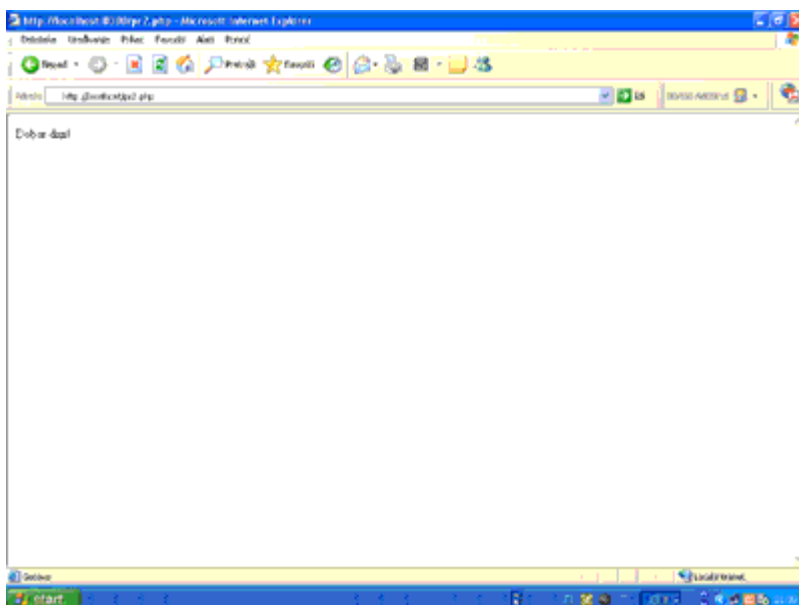
```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
```

```
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1250" />
</head>
<body>

<?php
echo "Dobar dan!";
?>

</body>
</html>
```

2. Pohranite dokument pod imenom pr2.php u mapu **C:\Program Files\xampp\htdocs**
3. Otvorite pr2.php u Web pregledniku pomoću adrese <http://localhost/pr2.php>



Objašnjenje moje druge PHP aplikacije

1. PHP kôd pišemo unutar HTML kôda.
2. PHP kôd je otok unutar HTML dokumenta.
3. PHP se izvršava na Apacheu.
4. Rezultat tog izvršavanja zajedno s preostalim HTML kôdom, Apache šalje Web pregledniku.
5. U Web preglednik će uvijek stići HTML dokument bez PHP kôda.

6. U gornjoj PHP aplikaciji smo upoznali ključnu riječ "echo". Isti rezultat bismo dobili i s ključnom riječi "print".
7. "echo" i "print" u PHP jeziku ispisuju na ekranu ono što smo naveli unutar navodnika.
8. Rezultat gornje PHP aplikacije je pojava Dobar dan! na ekranu.
9. Pogledate li izvorni kôd dobijene stranice vidjet ćete HTML kôd i Dobar dan!.
10. Rezultat gornje PHP aplikacije, otoka unutar HTML dokumenta, je: Dobar dan!
11. Apache je umjesto php programa poslao web pregledniku rezultat tog programa. Tako web stranice na našim ekranima uvijek sadrže rezultate a ne i programski kôd PHP-a koji je stvorio te rezultate.

Zadaci

Zadatak 1.

1. Provjerite postoji li na vašem računalu usluga IIS Admin. Ako postoji, zaustavite je.
2. Ponovo pokrenite uslugu IIS Admin.
3. Ponovo zaustavite uslugu IIS Admin.

Zadatak 2.

1. Instalirajte Apache i PHP.
2. Koristite XAMPP for Windows.

Zadatak 3.

1. Napišite, pohranite i isprobajte u radu svoj prvi PHP program:

```
<?php
phpinfo();
?>
```

Zadatak 4.

1. Isprogramirajte, pohranite i isprobajte u radu svoju drugu .php stranicu:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1250" />
```

```
</head>
<body>
<?php
echo "Dobar dan!";
?>

</body>
</html>
```

Test

Zaokružite točan odgovor:

1. IIS je
 - a. Web poslužitelj otvorenog kôda
 - b. Microsoftov Web poslužitelj
2. `phpinfo()` je jedna instrukcija.
 - a. Da.
 - b. Ne.
3. Funkcija `phpinfo()` se izvršila na Apache-u.
 - a. Ne.
 - b. Da.
4. Apache je poslao našem Web pregledniku rezultat funkcije `phpinfo()`.
 - a. Ne.
 - b. Da.
5. Rezultat funkcije `phpinfo()` je impresivni HTML dokument.
 - a. Da.
 - b. Ne.
6. PHP pišemo unutar HTML kôda.
 - a. Da.
 - b. Ne.
7. PHP je otok unutar HTML dokumenta.
 - a. Da.
 - b. Ne.
8. PHP se u našem primjeru izvršava na Apacheu.
 - a. Ne.
 - b. Da.
9. Stranice koje imaju nastavak `.PHP` u web preglednik stižu kao čiste HTML stranice bez PHP kôda.

- a. Da.
- b. Ne.

PHP varijable

Tipovi PHP varijabli

PHP ima sedam tipova varijabli. To su: string, integer, float, boolean, array, objekt i resurs.

String sadrži znakove i ima svojstvo teksta bez obzira o kojim se znakovima radilo.

Integer sadrži cijele brojeve, negativne i pozitivne.

Float sadrži decimalne brojeve i cijele brojeve.

Boolean sadrži 1 ili 0, samo ta dva broja koja zamjenjuju ključne izraze "true" - istina(1) i "false" - nije istina (0).

Array je specijalna varijabla po tome što može sadržavati više vrijednosti istovremeno. To su članovi arraya i razlikuju se po ključu (imenu) i indeksu (0, 1, 2, ...).

Objekt je još specijalnija varijabla od arraya u toliko što pored više članova istovremeno, još sadrži i svoje funkcije (metode) .

Resurs je sve što nije PHP podatak. Na primjer:

```
$x=mysql_query("SELECT * FROM dobavljac");
```

Varijabla \$x je ovdje dobila sve podatke iz tablice 'dobavljac', ali oni nisu u strukturi array, niti bilo kojoj drugoj koja bi se mogla odmah koristiti. Zato je varijabla \$x u ovoj instrukciji pohranila resurs. Tek će funkcija `mysql_fetch_array()` od resursa \$x napraviti asocijativni array, o kojem će biti puno riječi u nastavku knjige.

```
$x=mysql_query("SELECT * FROM dobavljac");  
while($redak = mysql_fetch_array($x))
```

U ovoj knjizi ćemo se dobro upoznati sa svim ovim varijablama.

Variable 1 + 2 = 3

1. U Notepadu tipkajte ovu php stranicu:

```
<!DOCTYPE html  
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html>  
<head>  
<title>PHP varijable</title>  
<meta http-equiv="Content-Type" content="text/html;  
charset=windows-1250" />  
</head>  
<body>  
<?php  
$x = 1;
```

```
$y = 2;  
echo $x + $y;  
?>  
</body>  
</html>
```

Pohranite pod nazivom **pr3.php** u mapu **htdocs**. Otvorite pr3.php u Web pregledniku preko adrese **<http://localhost/pr3.php>**

Otvorit će se web stranica na kojoj piše **3**.

Objašnjenje 1 + 2 = 3

U prethodnom zadatku smo tipkali ovaj PHP program:

```
<?php  
$x = 1;  
$y = 2;  
echo $x + $y;  
?>
```

Ovaj PHP program ima tri instrukcije. Svaka instrukcija završava s točkom zarez i radi lakšeg održavanja programa pisana je u zaseban redak. To je dobra programerska praksa.

Apache izvršava PHP program redom odozgo prema dolje, instrukciju po instrukciju.

U prvoj instrukciji smo deklarirali i inicijalizirali varijablu *x*. U PHP programu varijable deklariramo tako što im ispred imena pišemo znak dolara \$. Inicijalizirati varijablu znači dodijeliti joj vrijednost, što činimo pomoću znaka jednakosti i ispisa vrijednosti na desnoj strani tog znaka. Kad vrijednost na desnoj strani pišemo bez navodnih znakova " " znači da je vrijednost broj. S navodnim znakovima bi vrijednost bila string (tekst).

U prvoj instrukciji uvodimo u program varijablu *\$x* i dodijeljujemo joj vrijednost broj 1.

U drugoj instrukciji uvodimo u program varijablu *\$y* i dodijeljujemo joj vrijednost broj 2.

U trećoj instrukciji pomoću PHP ključne riječi *echo*, razmaknice, pa ispisa *\$x + \$y* želimo da PHP ispiše na ekran zbroj dviju varijabli. Kao i mi u matematici, tako i PHP za zbrajanje koristi znak +.

To je objašnjenje onog što se događalo u glavi programera dok je pisao ovaj program od tri instrukcije.

Što se događa na server računalu i serveru Apache za vrijeme izvršavanja ovog PHP programa?

U prvoj instrukciji je Apache zauzeo jedno mjesto u memoriji (RAM-u) servera, imenovao to mjesto s *\$x* i na tom mjestu pohranio vrijednost 1.

U drugoj instrukciji je Apache zauzeo još jedno mjesto u memoriji (RAM-u) servera, imenovao to mjesto s *\$y* i na tom mjestu pohranio vrijednost 2.

U trećoj instrukciji je Apache najprije pročitao ključnu riječ 'echo' i znao da ono što slijedi treba ispisati u HTML dokument. Što slijedi u instrukciji iza echo? Slijedi \$x i on to ime traži u memoriji RAM-u, pronalazi i uzima vrijednost koju u toj "ladici" zatiče. Zatim slijedi +. Apache prepoznaje taj znak matematičke operacije zbrajanja, uzima ga u "svoju memoriju" i ide dalje. Nailazi na varijablu \$y, traži je u RAM-u, i pronalazi u istoimenoj "ladici" vrijednost 2, uzima tu vrijednost 2 u svoju memoriju. U Apachevoj memoriji, a to je RAM servera, se posložila cijela treća instrukcija. Apache razumije tu instrukciju i izvršava je. Zbraja 1 i 2. Rezultat tog izvršavanja je "ispiši 3 u HTML dokument". To se i događa. HTML stranica dobiva 3 u sekciju <body>. Program je izvršen. Apache je naišao na znak ?>. Taj znak Apacheu govori "program je završen". Što Apache učini nakon tog znaka? Prestane izvršavati instrukcije jer ih nema. To je kraj tog PHP programa, ali ne i kraj fizičkog dokumenta na kojem se taj program nalazi.

Još uvijek su u RAM-u GLOBALNE varijable \$x i \$y:

```
$x = 1;
$y = 2;
```

Tek nakon </html>, PHP obriše varijable \$x i \$y.

```
?>
</body>
</html>
```

Na kraju izvršenja PHP programa, Apache ima web stranicu. To je HTML dokument kojeg šalje Web pregledniku, i kojeg možete vidjeti u izvornom kôdu pomoću naredbe View/Source ili Prikaz/Izvor:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP varijable</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
3
</body>
</html>
```

Na mjestu PHP programa sad je samo broj 3. Nema PHP kôda. Nikad ga neće ni biti. Posjetitelji naših PHP stranica nikad neće vidjeti naš PHP kôd. On 'leži' na serveru u svojoj datoteci i čeka sljedeći poziv.

Koliko žive varijable, a koliko živi program

1. U Notepadu tipkajte ovu web stranicu:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
```



```
<head>
<title>PHP varijable</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<p>Ovo što slijedi je PHP "prvi otok":</p>
<?php
$x = 1;
$y = 2;
echo $x + $y;
?>
<p>A sad ide PHP "drugi otok":</p>
<?php
$x = 10;
$y = 20;
echo $x + $y;
?>
</body>
</html>
```

2. Pohranite je pod nazivom pr4.php u mapu htdocs. Otvorite pr4.php u Web pregledniku preko adrese <http://localhost/pr4.php> i dobit ćete ovu web stranicu:

Ovo što slijedi je PHP "prvi otok":

3

A sad ide PHP "drugi otok":

30

Objašnjenje

Globalne varijable

U prvom PHP otoku u RAM-u su kreirane dvije varijable \$x i \$y. Dodijelili smo im vrijednosti 1 i 2. One su odmah kao takve GLOBALNE, što znači da će biti prisutne u RAM-u dok god traje PROCESUIRANJE web stranice.

U drugom PHP otoku smo istim globalnim varijablama koje već postoje u RAM-u, promijenili vrijednosti u 10 i 20.

Cijela web stranica, zato što ima nastavak .php, prolazi kroz INTERPRETER. Pri tom, XHTML ostaje ne promijenjen, a INTERPRETER ima posla samo u OTOCIMA PHP tj. ima posla između graničnika <?php ?>.

Varijable \$x i \$y nisu uništene na kraju prvog otoka. Ne. One su GLOBALNE a to znači da će biti u RAM-u do zadnjeg retka web stranice koja prolazi kroz INTERPRETER. Drugim riječima, bit će na raspolaganju SVIM PHP 'otocima' koji postoje na web stranici.

Primjer:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP varijable</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<p>Ovo što slijedi je PHP "prvi otok":</p>
<?php
$x = 1;
$y = 2;
echo $x + $y;
?>
<p>A sad ide PHP "drugi otok":</p>
<?php
echo $x + $y;
?></body>
</html>
```

U pregledniku ćemo dobiti ovo:

```
Ovo što slijedi je PHP "prvi otok":
3
A sad ide PHP "drugi otok":
3
```

Lokalne varijable

To su varijable unutar funkcija.

```
<?php
$a = 1;

function Test()
{
    echo $a;
}

Test();
?>
```

U web pregledniku ćemo dobiti prazan list. Varijabla \$a je globalna, ali se ta globalnost ne odnosi na unutrašnjost funkcije. Zato pozvana funkcija Test() nije napisala 1.

Postoji array \$GLOBALS koji živi cijelo vrijeme procesuiranja. On se kreira automatski i prihvaća kao svoje članove SVE varijable koje smo kreirali unutar graničnika <\$php ... ?>, osim varijabli koje kreiramo unutar funkcija. Svaki takav član dobije svoje indexe počevši od 0, 1 pa dalje, ali dobije i ključeve koji su jednaki imenima varijabli.

Tako je varijabla \$a u arrayu \$GLOBALS['a'].

Varijable unutar funkcija su lokalne i žive samo koliko traje procesuiranje funkcije.

Dohvat globalne varijable iz funkcije

```
<?php
$a = 1;

function Test()
{
    echo $GLOBALS['a'];
}

Test();
?>
```

Sad će nam preglednik izbaciti '1'. Ako unutar funkcije želimo dohvatiti globalnu varijablu, pisat ćemo ovako `$GLOBALS['a']`.

Dakle, postoji "životni vijek varijabli", ali također i životni vijek programa. Jer, kroz RAM nisu prošle samo varijable, kroz RAM su prošle sve instrukcije programa. Kao rijeka ponornica. Kao naslagani listovi koje netko uzima, list po list, pročita i ubacuje u stroj za mljevenje. Tko to uzima instrukciju po instrukciju, izvrši je i uništi? Procesor. Kad procesor uzme iz RAM-a posljednju instrukciju programa, izvrši je i uništi, to je i kraj života tog programa. Ništa njegovo više nema u RAM-u, kao da nikada nije ni bilo.

Čovjek se s programom suočava na dva načina. Prvi način je onaj kad čovjek piše program, instrukciju po instrukciju. Drugi način je onaj kad program radi.

Forma

1. U Notepadu tipkajte ovu web stranicu, imenujte je s `pr5.htm` i pohranite u mapu `htdocs`.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP varijable</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<form action="zbroji.php" method="POST">
<input type="text" name="x" /> +
<input type="text" name="y" />
<input type="submit" value="Izračunaj" />
</form>
</body>
</html>
```

To je u cijelosti HTML stranica, bez PHP kôda, pa smo je imenovali s nastavkom `.htm`.

2. Otvorite je u Web pregledniku pomoću adrese `http://localhost/pr5.htm` i vidjet ćete ovu formu:

<input type="text"/>	+	<input type="text"/>	<input type="button" value="Izračunaj"/>
----------------------	---	----------------------	--

Forma još neće zbrajati brojeve koje unesete jer nismo napravili PHP program koji će obaviti taj posao zbrajanja. U sljedećoj nastavnoj jedinici ćemo napraviti taj PHP program unutar web stranice koju ćemo nazvati `zbroji.php` jer se ta stranica poziva u prvom retku forme:

```
<form action="zbroji.php" method="POST">
```

Varijable izvan PHP

1. U Notepadu tipkajte ovu web stranicu, imenujte je `zbroji.php` i pohranite u mapu `htdocs`.

Napomena: kutne zagrade se tipkaju pomoću ALT+91 i ALT+93. Drži se pritisnutom lijeva tipka ALT i desnom se rukom tipka 91 na desnom brojčanom dijelu tipkovnice.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP varijable</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
Rezultat je:
<?php
echo $_POST["x"] + $_POST["y"];
?>
</body>
</html>
```

2. Sad otvorite u Web pregledniku web stranicu `pr5.htm` pomoću adrese `http://localhost/pr5.htm`. Tipkajte u prvo polje 1 u drugo 2 pa klik na Izračunaj.
3. Otvori se stranica `http://localhost/zbroji.php` na kojoj piše Rezultat je: 3

Objašnjenje

HTTP protokol ima svoje funkcije (metode) pomoću kojih prenosi vrijednosti iz FORME u RAM servera, kao varijable, pristigle izvan PHP kôda i stavljene na raspolaganje tom PHP kôdu. To su metode GET i POST.

U formi smo jedno polje imenovali s `x` a drugo s `y`. Kad smo ispunili polje `x` s 1 i polje `y` s 2 i pritisnuli na Izračunaj, metoda POST, koja je također navedena u formi, je prenijela te podatke na server Apache zajedno s pozivom stranice

zbroji.php. Kamo je stavila podatke $x=1$ i $y=2$? Stavila ih je u RAM servera. Jednu "ladicu" u memoriji servera je nazvala `$_POST["x"]` i u tu "ladicu" stavila broj 1. Drugu "ladicu" u memoriji servera je nazvala `$_POST["y"]` i u nju je stavila broj 2.

Nakon toga je pozvala stranicu zbroji.php. Kad netko pozove stranice koje imaju nastavak .php, u ovom slučaju taj netko je metoda POST, onda Apache zna po nastavku .php da tu ima i za njega posla. Počne pregledavati stranicu od početka prema kraju i propušta sve kako i zatekne, osim dijela koji se nalazi unutar graničnika `<?php i ?>`. Zna da su tu instrukcije PHP programa koje treba izvršiti.

```
<?php
echo $_POST["x"] + $_POST["y"];
?>
```

U ovom slučaju je Apache našao samo jednu instrukciju koja ispisuje na web stranicu zbroj varijabli `$_POST["x"] + $_POST["y"]`. Apache zbroji te varijable. Nakon toga Apache doda broj 3 na stranicu i pošalje je prema Web pregledniku. Cijela stranica je pregledana, na njoj nema više PHP otoka.

Varijable: Zadaci

Zadatak: 1

1. Napiši u Notepadu HTML stranicu na kojoj je forma kao u primjeru 5.
2. Napravi sljedeće promjene:
3. Neka stranica na kojoj je forma poziva samu sebe. Zato učini sljedeće.
4. Pohrani dokument s nastavkom .php, na primjer pr7.php, a ne kao tamo s nastavkom .htm.
5. U retku `<form action="pr7.php" method="POST">` napiši da je `action="pr7.php"` što znači: kad pritisnemo gumb "Izračunaj", bit će pozvana opet ista stranica pr7.php na kojoj je i forma.
6. Dodaj sljedeći PHP program i formu:

```
<form action="pr7.php" method="POST">
<input type="text" name="x" /> +
<input type="text" name="y" />
=
<?php
echo $_POST["x"] + $_POST["y"];
?>
```

```
<input type="submit" value="Izračunaj" />
</form>
```

7. Pohrani promjene. Otvori web stranicu pr7.php u Web pregledniku pomoću adrese `http://localhost/pr7.php`
8. Što ćeš vidjeti?
9. Upiši 1 u prvo polje, 2 u drugo polje i pritisni Izračunaj. Što ćeš sad vidjeti?

Rješenje

U prvom otvaranju stranice ću vidjeti:

+ = 0

Nakon pritiska na Izračunaj vidjet ću ovo:

+ = 3

Variable: Test

Zaokruži točan odgovor:

1. String je tekst.
 - a. Da.
 - b. Ne.
2. "12" je
 - a. string
 - b. broj
 - c. integer
3. Integer sadrži cijele brojeve, pozitivne i negativne.
 - a. Da.
 - b. Ne.
4. Boolean sadrži 1 ili 0.
 - a. Da.
 - b. Ne.
5. Lokalna varijabla živi koliko živi i njena funkcija.
 - a. Da.
 - b. Ne.
6. Globalna varijabla živi od prvog retka HTML dokumenta (web stranice) do zadnjeg retka web stranice.

- a. Da.
 - b. Ne.
7. Varijable prepoznamo po znaku '\$' ispred imena.
- a. Da.
 - b. Ne.

PHP uvjetni programski kôd

IF....ELSE

1. U Notepadu tipkajte ovaj php dokument i pohranite pod imenom pr8.php.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP varijable</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<form action="pr8.php" method="POST">
<input type="text" name="x" /><br />
<?php
if ($_POST["x"] < 20)
{
print "Vi imate manje od 20 godina.";
}
else
{
print "Vi nemate manje od 20 godina.";
}
?>
<input type="submit" value="U redu" />
</form>
</body>
</html>
```

2. Otvorite pr8.php u Web pregledniku pomoću adrese
<http://localhost/pr8.php> i isprobajte u radu.

Objašnjenje

Struktura IF se sastoji od:

1. prve instrukcije koja počinje s ključnom riječi IF, razmaknica, otvorena zagrada, TVRDNJA, zatvorena zagrada. Iza ove instrukcije ne stavljamo točku zarez ;,

2. unutar vitičastih zagrada pišemo one instrukcije koje će se dogoditi AKO je TVRDNJA istinita (true ili 1),
3. u našem slučaju smo imali samo jednu instrukciju koja će ispisati na web stranici "Vi imate manje od 20 godina.",
4. ključna riječ 'ELSE' prethodi instrukcijama koje će se izvršiti ako TVRDNJA nije točna (false ili 0),
5. u našem slučaju smo imali samo jednu instrukciju koja će ispisati na web stranici "Vi imate više od 20 godina.".

Zadaci

Zadatak: 1

3. Napišite PHP stranicu s formom u kojoj ćemo upisati neko ime.
4. Neka se poziva ista stranica nakon pritiska na gumb SUBMIT.
5. Nakon upisa imena u polje INPUT i pritiska na SUBMIT, neka se na stranici ispiše
6. "Dobro došao Ivica!" ako ste tipkali ime Ivica.
7. "Vi niste Ivica." ako ste tipkali bilo koje ime osim imena Ivica.

Test

Zaokruži točan odgovor.

1. Instrukcija koja počinje s IF uvijek u okruglim zagradama sadrži neku TVRDNJU.
 - a. Da.
 - b. Ne.
2. Kad će se izvršiti instrukcije ispod retka IF?
 - a. Ako je tvrdnja istinita (true ili 1).
 - b. Ako je tvrdnja laž (false ili 0).
3. Kad će se izvršiti instrukcije ispod ELSE?
 - a. Ako je tvrdnja istinita (true ili 1).
 - b. Ako je tvrdnja laž (false ili 0).

Logički i usporedni operatori

Logički i usporedni operatori se najčešće koriste u strukturama IF, pa je sad pravi trenutak za upoznati se s njima.

Evo logičkih operatora u PHP-u:

Operator	Opis	Primjer
&&	i	x=20 y=18

		(x < 21 && y > 14) rezultat je true (1)
	ili	x=20 y=18 (x==10 y==12) rezultat je false (0)
!	ne	x=20 y=18 !(x==y) rezultat je true (1)

Usporedni operatori u PHP-u:

Operator	Opis	Primjer
==	ima istu vrijednost kao	15==18 rezultat je false (0)
!=	nema istu vrijednost kao	15!=18 rezultat je true (1)
>	je veće od	15>18 rezultat je false (0)
<	je manje od	15<18 rezultat je true (1)
>=	je veće ili jednako od	15>=18 rezultat je false (0)
<=	je manje ili jednako od	15<=18 rezultat je true (1)

Tablica istine za logičko 'i'

A	B	C = A i B	C
0	0	A && B	0
0	1	A && B	0
1	0	A && B	0
1	1	A && B	1

Tablica istine za logičko 'ili'

A	B	C = A ili B	C
0	0	C = A B	0
0	1	C = A B	1
1	0	C = A B	1
1	1	C = A B	1

Napomena: okomitu crtu | tipkajte pomoću alt+124 (lijevi alt i 124 na desnom brojevnom dijelu tipkovnice).

Zadaci iz logičkih i usporednih operatora

1. Napravite php stranicu koja će imati dva polja: korisničko ime i zaporka. Ako upišete 'ivica' i '1234' vratit će vam se dobrodošlica 'Dobro nam se

vratili Ivica!'. U suprotnom ćete dobiti poruku 'Molimo ponovite, korisničko ime ili zaporka su vam pogrešno unijeti.'

Rješenje

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP logičko i</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<form action="zadatak1.php" method="POST">
Korisničko ime: <input type="text" name="korisnik" /><br />
Zaporka: <input type="password" name="zaporka" /><br />
<input type="submit" value="Izračunaj" />
</form>

<?php
if ($_POST['korisnik'] == 'ivica' && $_POST['zaporka'] == '1234')
    print "Dobro nam se vratili Ivica!";
else
    print "Molimo ponovite, korisničko ime ili zaporka su vam
    pogrešno unijeti.";
?>

</body>
</html>
```

2. A sad želite propustiti više osoba prema svojoj stranici index.php. Naredba za preusmjeravanje je: header("Location: index.php");, ako je stranica index.php u istoj mapi u kojoj i stranica zadatak2.php.

Rješenje

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP logičko i / ili</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<form action="zadatak2.php" method="POST">
Korisničko ime: <input type="text" name="korisnik" /><br />
Zaporka: <input type="password" name="zaporka" /><br />
<input type="submit" value="Izračunaj" />
</form>

<?php
if ($_POST['korisnik'] == 'ivica' && $_POST['zaporka'] == '1234' or
    $_POST['korisnik'] == 'ivana' && $_POST['zaporka'] == '56zz' or
```

```

$_POST['korisnik'] == 'vedran' && $_POST['zaporka'] == '89kl')
{
    header("Location: index.php");
    exit;
}
else
    print "Molimo ponovite, korisničko ime ili zaporka su vam
pogrešno unijeti.";
?>
</body>
</html>

```

Napomena: Kad je samo jedna instrukcija ispod tvrdnje IF, onda nije potrebno pisati vitičaste zagrade. Ako je više od jedne instrukcije onda je pisanje vitičastih zagrada obvezno. Isto vrijedi i za instrukcije ispod 'else'.

Objašnjenje logičkih operatora u gornjim primjerima

```
if ($_POST['korisnik'] == 'ivica' && $_POST['zaporka'] == '1234')
```

Tvrdnja uvijek mora biti jednaka 1 (true) da se izvedu instrukcije ispod IF tvrdnje. U suprotnom, kad je tvrdnja jednaka 0 (false), procesor izvršava instrukcije ispod else.

Čemu je jednaka gornja tvrdnja ako tipkamo 'ivica' i '1234'? Postavimo sebi jedno pod pitanje: ima li u gornjoj tvrdnji možda i pod-tvrdnji? Ima. To su ove dvije pod-tvrdnje:

pod tvrdnja	što se tvrdi
<code>\$_POST['korisnik'] == 'ivica'</code>	globalna varijabla je jednaka ivica
<code>\$_POST['zaporka'] == '1234'</code>	globalna varijabla je jednaka 1234

Čemu je jednaka ukupna tvrdnja ako tipkamo 'ivica' i '1234'?

pod tvrdnja	čemu je jednaka
<code>\$_POST['korisnik'] == 'ivica'</code>	1 (istina ili true)
<code>\$_POST['zaporka'] == '1234'</code>	1 (istina ili true)
Na osnovu gornjeg, ukupna tvrdnja je: <code>1 && 1</code>	1

Ako pogledate tablicu istine za logičko 'i' vidjet ćete da `1 && 1` daju 1, što znači da je ukupna tvrdnja jednaka 1, što dalje znači izvršit će se instrukcije ispod tvrdnje IF.

Zaključak: Unutar zagrade može biti puno tvrdnji koje su povezane logičkim operatorima. Bez obzira koliko tvrdnji bilo u zagradi, logički rezultat svih tih tvrdnji zajedno je uvijek 1 ili 0. Ako je jedan izvršavaju se naredbe ispod IF. Ako je 0 izvršavaju se naredbe ispod ELSE.

Petlje

For

1. U Notepadu tipkajte ovaj php dokument i pohranite pod imenom pr9.php.

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP varijable</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<?php
for ($i = 1; $i < 100; $i = $i + 2)
{
print "$i";
}
?>
</body>
</html>
```

2. Otvorite pr9.php u Web pregledniku pomoću adrese <http://localhost/pr9.php> i isprobajte u radu. Dobili ste ispis brojeva 135....99.

Objašnjenje

Petlja FOR se sastoji od:

-prve instrukcije koja počinje s ključnom riječi FOR, razmaknica, otvorena zagrada, tri argumenta, zatvorena zagrada. Iza ove instrukcije ne stavljamo točku zarez ;. Argumenti su odijeljeni ; točkom zarez. U prvom argumentu kreiramo varijablu \$i i iniciramo joj početnu vrijednost 1. U drugom argumentu određujemo gornju granicu varijable \$i < 100, a to je 99, pošto se prolazi petlje broje s cijelim brojevima. Zato svi i dajemo ovoj varijabli ime \$i da nas podsjeti da je riječ o tipu varijable integer. U trećem argumentu određujemo u kojim koracima će se mijenjati vrijednost varijable \$i. U našem slučaju je to broj 2. Prvi prolaz \$i=1, drugi prolaz &i=3 itd.

1. Unutar vitičastih zagrada pišemo one instrukcije koje će se ponavljati.
2. U našem slučaju će se svaki put ispisati vrijednost varijable \$i.

FOR: Zadaci

Zadatak: 2

1. Pomoću petlje FOR ispišite 10 slova A: AAAAAAAAAA

Test

Zaokružite točan odgovor.

2. Instrukcija koja počinje s FOR u zagradi ima tri argumenta.
 - a. Da.
 - b. Ne, ima samo jedan argument.
 - c. Ne, ima samo dva argumenta.
3. "Živi" li varijabla i nakon prestanka rada petlje i kako bismo to provjerili.
 - a. Da. Nakon posljednje vitičaste zagrade }, a prije finalnog graničnika ?>, napišemo ove dvije instrukcije

```
echo "$i";
exit;
```
 - b. Ne. Nakon posljednje vitičaste zagrade }, a prije finalnog graničnika ?>, napišemo ove dvije instrukcije

```
echo "$i";
exit;
```

While

Primjer 1.

```
<html>
<body><?php
$i=1;
while($i<=100)
{
    echo $i . "<br />";
    $i++;
}
?></body>
</html>
```

Izbacuje:

1

2

itd. do 100.

Objašnjenje:

```
echo $i . "<br />";
```

Kad dodajemo stringove jedne drugima koristi se točka a ne +. Oko varijable se ne stavljaju navodnici, samo oko stringova.

Primjer 2.

```
<html>
<body><?php
$i=1;
while($i<=6)
{
    echo "<h" . $i . ">Naslov</h" . $i . "<br />";
    $i++;
}
```

```
?></body>
</html>
```

Rezultat:

Naslov

Naslov

Naslov

Naslov

Naslov

Naslov

Zadatak: Kako glasi kôd koji će izbaciti obrnuto gornjem?

Rješenje:

```
<html>
<body><?php
$i=1;
$x=7;
while ($i<=6)
{
    $y=$x - $i;
    echo "<h" . $y . ">Naslov</h" . $i . "<br />";
    $i++;
}
?></body>
</html>
```

Rezultat:

Naslov

Naslov

Naslov

Naslov

Naslov

Naslov

Preporuka profesorima i učenicima

Zadržite se što je moguće duže na primjerima strukture IF, logičkim i usporednim operatorima i petljama. Crpite primjere iz svoje svakodnevnice i isprogramirajte ih. Vježbajte logičko && i logičko ili ||. Vježbajte slaganje informacija od više stringova i varijabli.

Zaključak nastavne cjeline 1

U prvoj nastavnoj cjelini smo instalirali PHP i Apache na svoje Windows računalo. Počeli smo od jedne jedine programske instrukcije, naučili prve ključne riječi print i echo. Nastavili smo s PHP varijablama. Naučili smo koliko žive varijable, a koliko žive programi. HTML forma nam je pomogla da upoznamo kako varijable pristižu izvan PHP. Nastavnu jedinicu smo završili s IF uvjetnom strukturom i petljama.

Na redu je pristup bazi podataka. Ali prije toga moramo se upoznati s najčešćim poslužiteljem baza podataka koji se danas koristi uz PHP – MySQL-om.

Nastavna cjelina 2



MySQL za početnike

Metodika -----

Instalacija

Instalacija se dogodila zajedno s Apachem i PHP, pomoću XAMPP za Windows. U mapi BIN se nalazi i program za komunikaciju s MySQL-om. Najbolje učenje SQL jezika današnjih baza podataka je upravo preko neredbodavnog retka:

```
mysql>
```

Satovi su puni vježbi. Svaki blok sat može biti kreacija jedne baze, tablica, unos podataka u tablice, brisanje tablica, brisanje baza.

Vježbe

Temeljne naredbe koje ćemo trebati u našim PHP stranicama za komunikaciju s bazom podataka su: SELECT, INSERT, DELETE i UPDATE.

Ovdje je prilika uvježbati i sve upite koji nam se pojavljuju u kasnijim primjerima CMS-a, e-trgovine i drugih.

MySQL je poslužitelj baza. Ili, MySQL je RDBMS (Relation Database Management System) ili sustav za upravljanje relacijskim bazama podataka.

Treba vježbati na što je moguće više primjera baza podataka s tablicama koje su u relaciji. Nakon što se prođe ova nastavna cjelina koja nas uvodi u MySQL i SQL skriptni jezik, preporučujem vježbanje na bazi koju koristimo za aplikaciju CMS. U podatkovnom objektu te aplikacije ima mnogo upita koji se baziraju na više tablica. Također, tu je i tražilica i FULL TEXT opcija traženja, što je odlično vježbati i na ovoj mysql> naredbodavnoj liniji.

phpMyAdmin

Instalacija pomoću paketa XAMPP vam je na računalo instalirala i izvanredni program php grafičkog sučelja za komunikaciju s MySQL poslužiteljem.

Preporučujem upoznavanje s ovim sučeljem, ali ne i korištenje u vježbama u kojima treba pisati SQL kod. A to su sve vježbe u ovoj knjizi.

Učenici koji žele više i imaju više interesa i sami će vrlo brzo napraviti svoje vlastite php stranice za komunikaciju s MySQL. Mislim na onu komunikaciju u kojoj pravimo nove baze, tablice, punimo te tablice itd.

Udžbenik-----

Što ćemo naučiti u ovoj nastavnoj cjelini?

Instalacija MySQL

Ako ste postupili kako je navedeno u poglavlju Instalacija PHP, onda se na vašem računalu već nalazi i MySQL. Provjeriti možete ovako:

1. Start/Upravljačka ploča (Control Panel)/Administrativni alati (Administrative Tools)/Servisi (Service)/
2. Ako je u popisu servisa i MySQL i ako piše 'started' u stupcu Status onda je MySQL spreman za vaš rad s njim.

Što je točno MySQL?

MySQL je samo jedna od mnogobrojnih usluga na našem računalu. To je software. Što nam omogućava taj software, ta usluga? MySQL poslužitelj nam omogućava kreiranje relacijskih baza podataka i njihovo održavanje.

Naša komunikacija s MySQL

S MySQL možemo komunicirati pomoću:

1. MySQL monitora, programa koji je instaliran na vašem računalu kad i MySQL,
2. PHP stranica koje ćemo sami kreirati u svojim PHP aplikacijama,
3. PHP stranica koje su drugi kreirali i stavili besplatno na raspolaganje svima koji to žele. Takva jedna PHP aplikacija je također već na vašem računalu. Instalirala se zajedno s MySQL i zove se phpMyAdmin. Otvarit će se u vašem web pregledniku ako tipkate u polje address:
<http://localhost:8080/phpmyadmin/> ili <http://localhost/phpmyadmin/>
4. Notepada. Ispišemo SQL naredbe u Notepadu i pohranimo, na primjer pod imenom 'unos.sql', u MySQL-ovu mapu 'bin'. Otvorimo mapu 'bin' u Command Promptu, mapu u kojoj je program MySQL monitor. Ako smo u Notepadu ispisali samo SQL naredbe za već postojeću bazu podataka, na primjer bazu 'skola', tipkat ćemo ovu naredbu:

```
bin>mysql -u root -D skola < unos.sql  
Enter
```

Ako smo u unos.sql ispisali i naredbe create database skola; use skola; itd. onda ćemo tipkati u Command Promptu ovu naredbu:

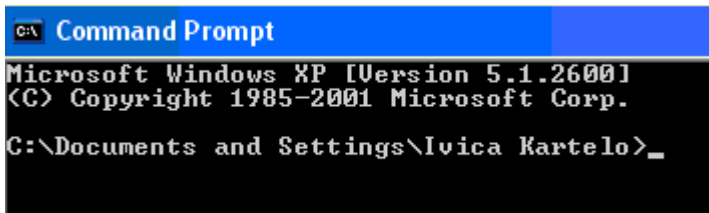
```
bin>mysql -u root < unos.sql  
Enter
```

Učiti MySQL znači učiti SQL, skriptni jezik baza podataka. SQL se najbolje uči pomoću MySQL monitora.

MySQL monitor

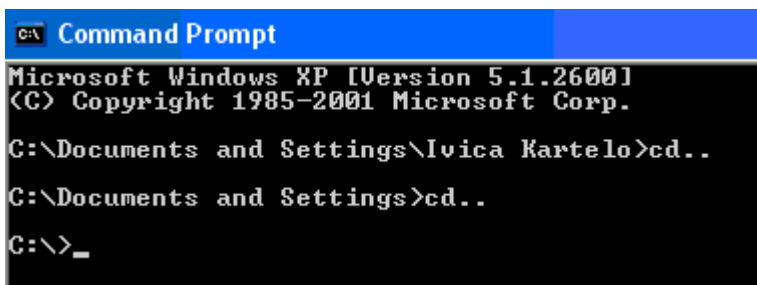
Pokretanje MySQL monitora preko Command Prompt

1. Start/Accessories/Command Prompt ili Start/Run/tipkajte cmd pa Enter.



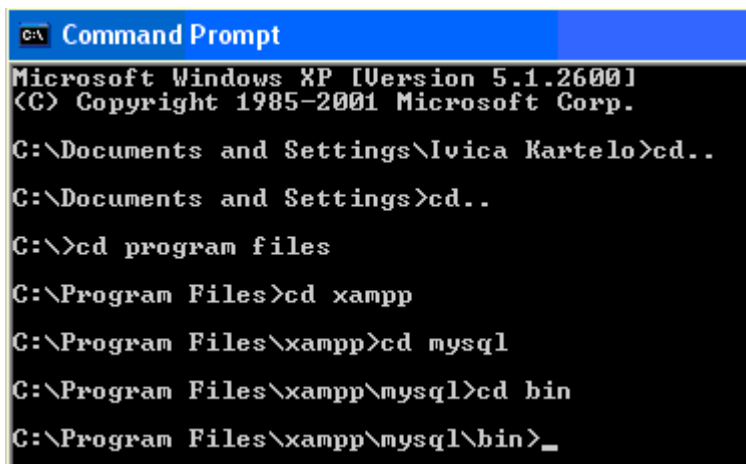
```
C:\ Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Ivica Kartelo>_
```

1. cd..
2. cd..



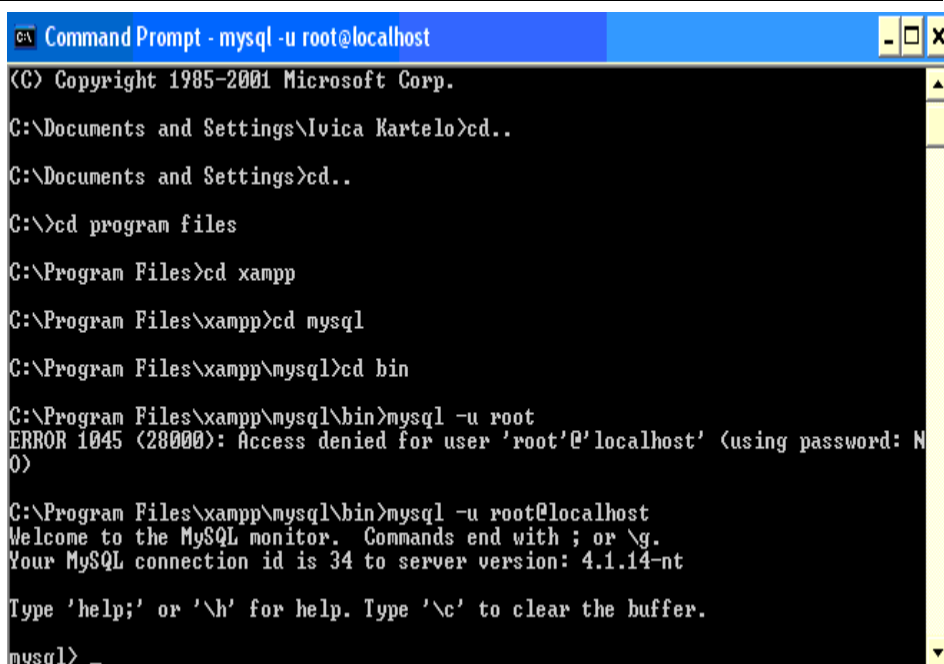
```
C:\ Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Ivica Kartelo>cd..
C:\Documents and Settings>cd..
C:\>_
```

3. cd program files
4. cd xampp
5. cd mysql
6. cd bin



```
C:\ Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Ivica Kartelo>cd..
C:\Documents and Settings>cd..
C:\>cd program files
C:\Program Files>cd xampp
C:\Program Files\xampp>cd mysql
C:\Program Files\xampp\mysql>cd bin
C:\Program Files\xampp\mysql\bin>_
```

7. mysql -u root



```
CA Command Prompt - mysql -u root@localhost
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Ivica Kartelo>cd..
C:\Documents and Settings>cd..
C:\>cd program files
C:\Program Files>cd xampp
C:\Program Files\xampp>cd mysql
C:\Program Files\xampp\mysql>cd bin
C:\Program Files\xampp\mysql\bin>mysql -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
C:\Program Files\xampp\mysql\bin>mysql -u root@localhost
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 34 to server version: 4.1.14-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Kao što vidite iz ovog ekrana, na mom računalu nije bilo dovoljno samo root već sam morao tipkati root@localhost. To je zato što sam ja posebno instalirao MySQL, a ne u kompletu XAMPP.

Mogli ste pristupiti i pomoću anonimnog korisnika:

```
bin>mysql
Enter
```

Došli ste do MySQL monitora, točnije MySQL naredbenog retka:

```
mysql>
```

Napomena: 'mysql -u root' je naredba za uspostavljanje veze s poslužiteljem MySQL.

-u.....user (korisnik)

U nastavku je objašnjenje inicijalnih korisnika.

Objašnjenje inicijalnih korisničkih imena MySQL-a

<http://mysql.com/doc/refman/5.0/en/default-privileges.html>

Napomena: Ukoliko samo učite na svom računalu SQL jezik, radite vježbe iz ove knjige i slično, najbolje ostavite inicijalne korisnike kako su inicijalno i postavljeni. Jer ako dodajete zaporke, možete ih zaboraviti. Naveo sam u nastavku adresu na kojoj je opisano kako možete vratiti korisnika root u inicijalno stanje.

Tijekom instalacije MySQL-a automatski se kreira i tablica s popisom inicijalnih korisnika. U toj tablici je korisnik 'root' bez zaporke. To je super korisnik koji ima sva prava.

Kreirana su i još dva anonimna korisnika. To su korisnici kojima je prazno i korisničko ime i zaporka. Jedan ima sva prava kao i korisnik root, a drugi ima sva prava pristupa bazama koje počinju s 'test'.

To znači da se iz naredbodavnog retka može ući i ovako u mysql poslužitelj:
kao anonimni korisnik:

```
....bin>mysql
```

ili kao root korisnik

```
....bin>mysql -u root
```

To znači da nakon inicijalne instalacije MySQL svatko može na vašem računalu prići MySQL poslužitelju sa svim pravima.

Ukoliko želite to promijeniti morate dodati zaporce inicijalnim korisnicima root i anonimnim ili ih potpuno ukloniti, i kreirati nove korisnike sa zaporkama.

Kreiranje zaporce za anonimne korisnike

```
bin> mysql -u root
mysql> SET PASSWORD FOR '@'localhost' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR '@'%' = PASSWORD('newpwd');
```

'localhost' je ime web poslužitelja na vašem lokalnom računalu.

Ime poslužitelja kao i svih inicijalnih korisnika možete vidjeti pomoću ove naredbe u mysql naredbodavnom retku:

```
mysql> SELECT Host, User FROM mysql.user;
```

```
mysql> select host, user from mysql.user;
+-----+-----+
| host      | user  |
+-----+-----+
| %         | %     |
| %         | admin |
| localhost | admin |
| localhost | ivica |
| localhost | root  |
+-----+-----+
6 rows in set (0.09 sec)
```

1. Sljedeće naredbe dodaju zaporku istovremeno svim anonimnim korisnicima (inicijalno su uspostavljena dva anonimna korisnika):

```
bin> mysql -u root
mysql> UPDATE mysql.user SET Password =
PASSWORD('nova_zaporka')
-> WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

2. Naredba FLUSH PRIVILEGES je potrebna kako bi MySQL odmah primijenio zaporce. U suprotnom bi to učinio tek nakon restartanja servera.
3. Potpuno uklanjanje anonimnih korisnika:

```
shell> mysql -u root
mysql> DELETE FROM mysql.user WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

4. Dodijeljivanje zaporce root korisniku:

```
bin> mysql -u root
```

```
mysql> SET PASSWORD FOR 'root'@'localhost' =  
PASSWORD('newpwd');  
mysql> SET PASSWORD FOR 'root'@'%' = PASSWORD('newpwd');
```

5. ili odjednom svim root korisnicima

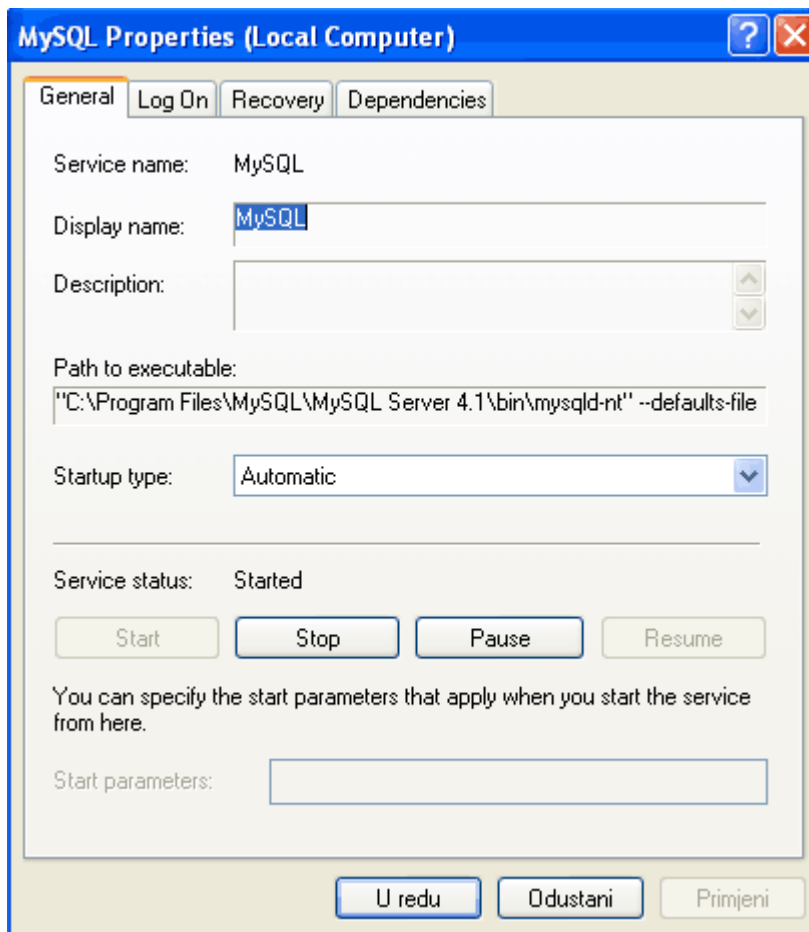
```
bin> mysql -u root  
mysql> UPDATE mysql.user SET Password = PASSWORD('newpwd')  
-> WHERE User = 'root';  
mysql> FLUSH PRIVILEGES;
```

6. Ako zaboravite zaporku i želite vratiti root kao što je bio, bez zaporkke, postupak je opisan na ovoj adresi:

<http://mysql.com/doc/refman/5.0/en/resetting-permissions.html>

7. Put do mape 'bin' do koje morate doći u Command Promptu ćete naći ovako:

- Start/Control Panel/Administrative Tools/Services
- Klik desnom tipkom miša na MySQL servis, pa odaberite Properties.



-
-
-
- d. U mom slučaju je put:

C:\Program Files\MySQL\MySQL Server 4.1\bin\

Ako želite kreirati novog korisnika, učinite to ovako:

<http://mysql.com/doc/refman/5.0/en/adding-users.html>

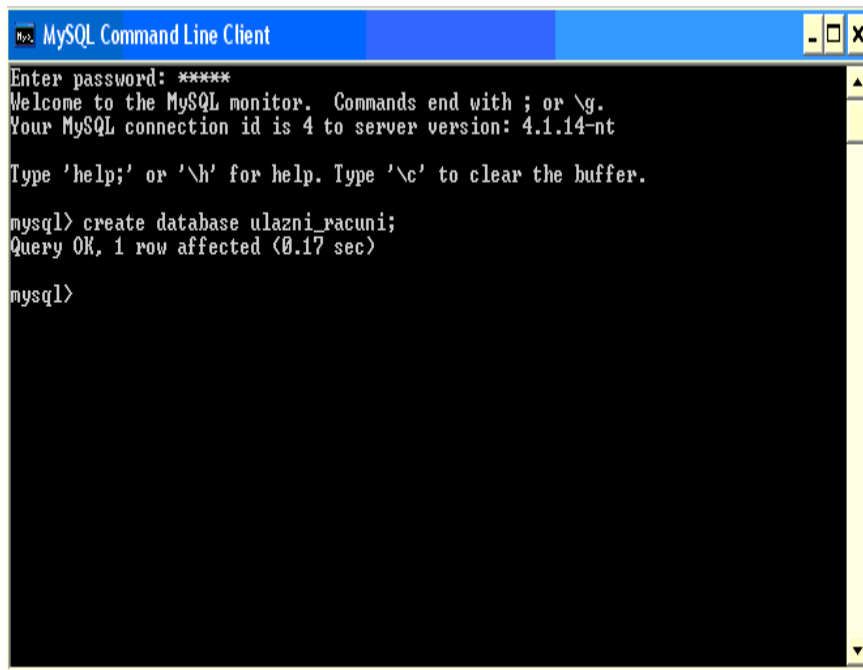
```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'ivica'@'localhost'  
-> IDENTIFIED BY 'neka_zaporka' WITH GRANT OPTION;
```

Kreiranje baze podataka 'ulazni_racuni'

1. Na MySQL naredbodavnom retku tipkam:

```
mysql>create database ulazni_racuni;
```

2. Enter.



```
MySQL Command Line Client  
Enter password: *****  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 4 to server version: 4.1.14-nt  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql> create database ulazni_racuni;  
Query OK, 1 row affected (0.17 sec)  
  
mysql>
```

Kreiranje tablica

Nastavljamo graditi našu bazu podataka ulazni_racuni, tako što ćemo toj bazi kreirati dvije tablice: dobavljac i racuni. Ne zaboravite na kraju svake SQL naredbe tipkati točku zarez ;.

3. Tipkajte naredbu:

```
mysql>use ulazni_racuni;
```

4. Enter. Tako ste došli u vezu s bazom podataka ulazni_racuni.

5. Tipkajte naredbu za kreiranje tablice dobavljac:

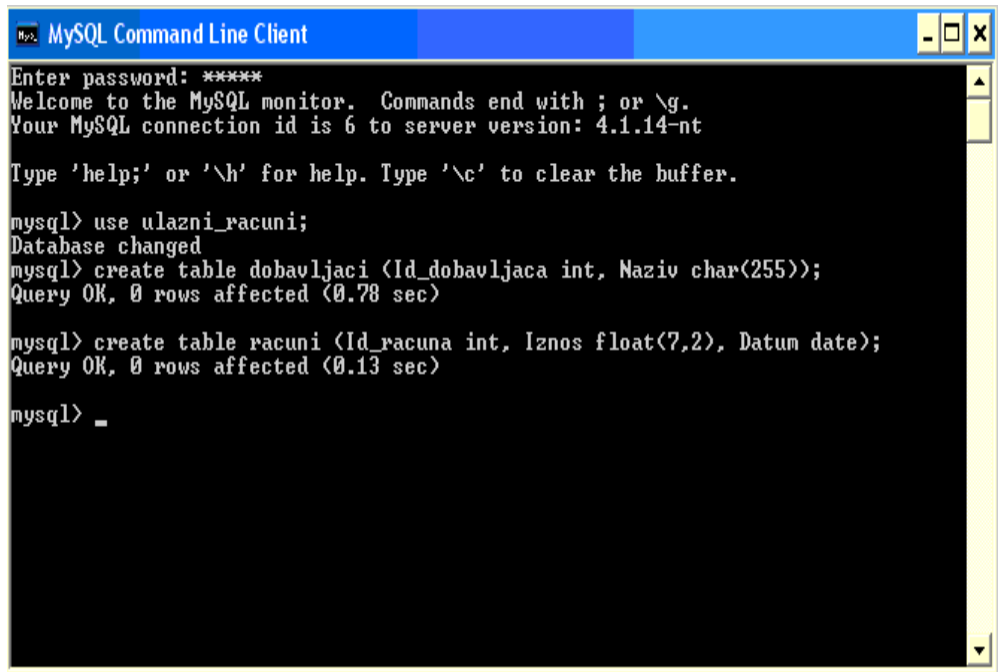
```
mysql>create table dobavljac (Id_dobavljacka int, Naziv  
char(255));
```

6. Enter.

7. Tipkajte naredbu za kreiranje tablice racuni:

```
mysql>create table racuni (Id_racuna int, Iznos float(7,2),  
Datum date);
```

8. Enter.



The screenshot shows a window titled "MySQL Command Line Client". The text inside the window is as follows:

```
Enter password: *****  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 6 to server version: 4.1.14-nt  
  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
  
mysql> use ulazni_racuni;  
Database changed  
mysql> create table dobavljac_i (Id_dobavl_jaca int, Naziv char(255));  
Query OK, 0 rows affected (0.78 sec)  
  
mysql> create table racuni (Id_racuna int, Iznos float(7,2), Datum date);  
Query OK, 0 rows affected (0.13 sec)  
  
mysql> _
```

Slijedi: Unos podataka u tablice.

Unos podataka u tablice

1. Otvorili ste MySQL konzolu i spojili se na MySQL pa na bazu ulazni_racuni ovako:

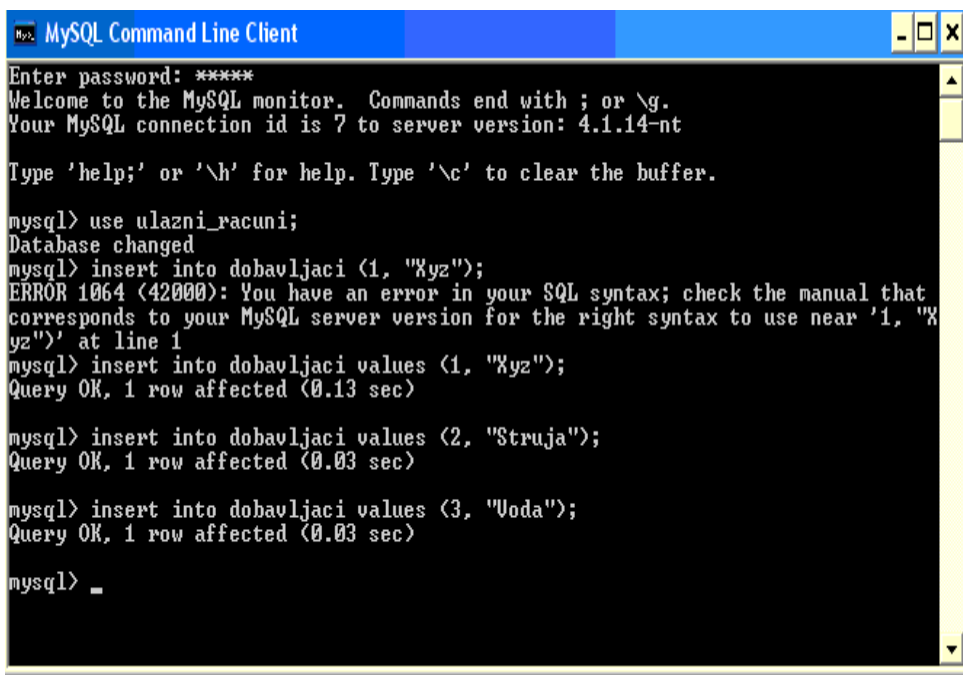
```
bin>mysql  
ENTER  
mysql>use ulazni_racuni;  
Enter
```

2. Tipkajte naredbu za unos podataka u prvi redak tablice dobavljac_i:

```
mysql>insert into dobavljac_i values (1, "Xyz");  
Enter.
```
3. Tipkajte naredbu za unos podataka u drugi redak tablice dobavljac_i:

```
mysql>insert into dobavljac_i values (2, "Struja");  
Enter
```
4. Tipkajte naredbu za unos podataka u treći redak tablice dobavljac_i:

```
insert into dobavljac_i values (3, "Voda"); Enter.
```



```
MySQL Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7 to server version: 4.1.14-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use ulazni_racuni;
Database changed
mysql> insert into dobavljac (1, "Xyz");
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near '1, 'X
yz'' at line 1
mysql> insert into dobavljac values (1, "Xyz");
Query OK, 1 row affected (0.13 sec)

mysql> insert into dobavljac values (2, "Struja");
Query OK, 1 row affected (0.03 sec)

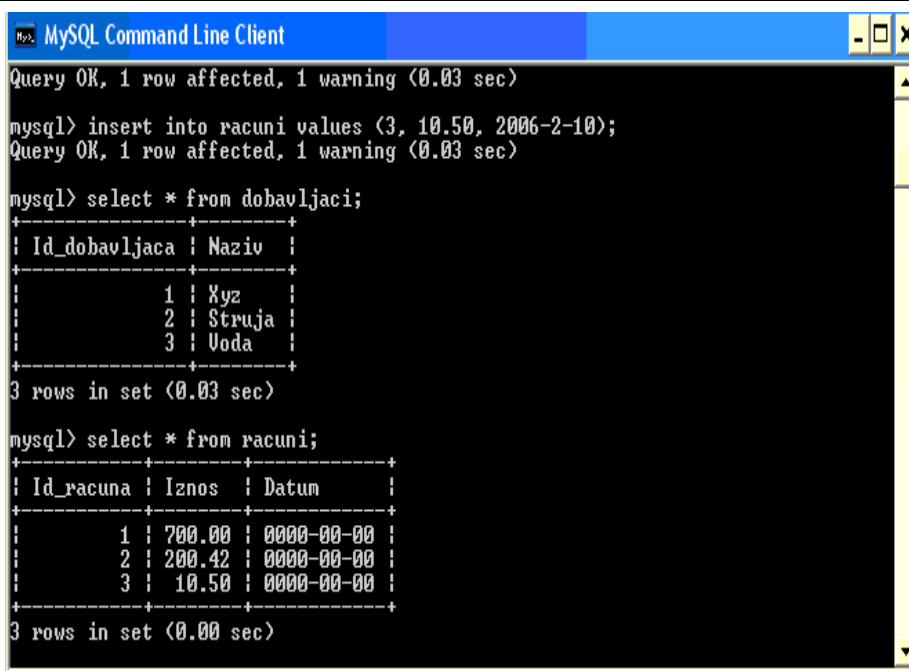
mysql> insert into dobavljac values (3, "Voda");
Query OK, 1 row affected (0.03 sec)

mysql> _
```

5. Tipkajte naredbu za unos podataka u prvi redak tablice racuni:
`mysql>insert into racuni values (1, 700, "2006-2-10");` Enter.
6. Tipkajte naredbu za unos podataka u drugi redak tablice racuni:
`mysql>insert into racuni values (2, 200.42, "2006-2-10");`
Enter.
7. Tipkajte naredbu za unos podataka u treći redak tablice racuni:
`mysql>insert into racuni values (3, 10.50, 2006-2-10);` Enter.

Pregled tablica

8. Tipkajte naredbu za pregled tablice dobavljac:
`mysql>select * from dobavljac;` Enter.
9. Tipkajte naredbu za pregled tablice racuni:
`mysql>select * from racuni;` Enter.



```
MySQL Command Line Client
Query OK, 1 row affected, 1 warning (0.03 sec)

mysql> insert into racuni values (3, 10.50, 2006-2-10);
Query OK, 1 row affected, 1 warning (0.03 sec)

mysql> select * from dobavljac;
+-----+-----+
| Id_dobavljac | Naziv |
+-----+-----+
| 1 | Kyz |
| 2 | Struja |
| 3 | Voda |
+-----+-----+
3 rows in set (0.03 sec)

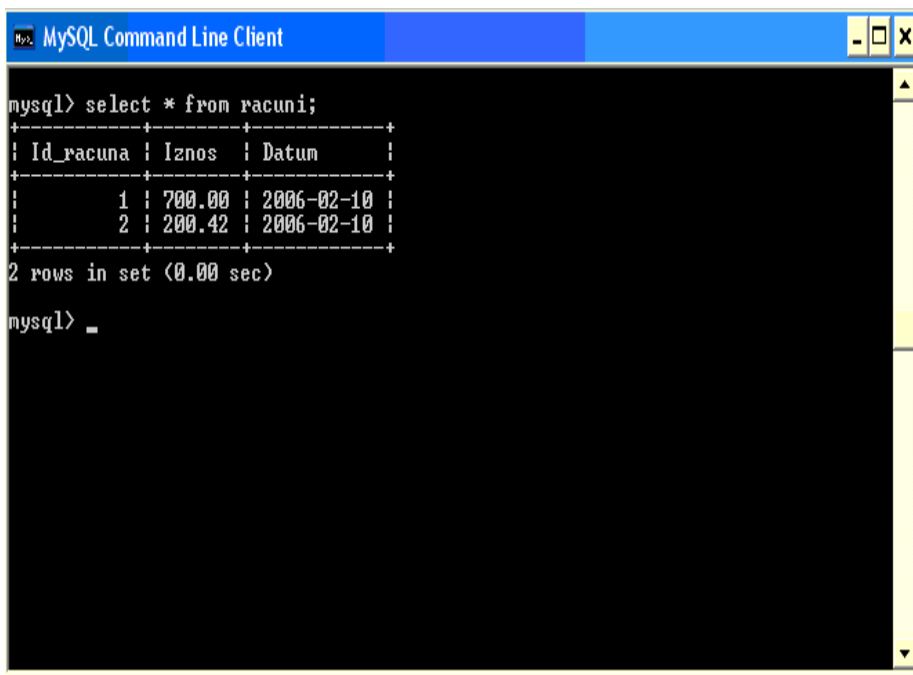
mysql> select * from racuni;
+-----+-----+-----+
| Id_racuna | Iznos | Datum |
+-----+-----+-----+
| 1 | 700.00 | 0000-00-00 |
| 2 | 200.42 | 0000-00-00 |
| 3 | 10.50 | 0000-00-00 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

Napomena: Datumi su neispravni jer ih nismo unijeli kao string (pod navodnicima).

Sljedi: Ispravljanje datuma u tablici racuni.

Ispravljanje datuma u tablici racuni

1. Spojite se na MySQL pa na bazu ulazni_racuni.
2. Tipkajte naredbu za ispravljanje datuma:
`mysql>update racuni set Datum = "2006-02-10" where Datum = "0000-00-00"; Enter.`
3. Tipkajte naredbu za brisanje trećeg retka u tablici racuni:
`mysql>delete from racuni where Id_racuna = '3'; Enter.`
4. Tipkajte naredbu:
`mysql>select * from racuni; Enter.`



The screenshot shows a window titled "MySQL Command Line Client". The command prompt shows the command `mysql> select * from racuni;` and the output is a table with three columns: `Id_racuna`, `Iznos`, and `Datum`. The table contains two rows of data. Below the table, it says "2 rows in set (0.00 sec)". The prompt then shows `mysql> _`.

Id_racuna	Iznos	Datum
1	700.00	2006-02-10
2	200.42	2006-02-10

Sljedi: Brisanje baze. Brisanje tablica.

Brisanje baze. Brisanje tablica

Brisanje baze i tablice isprobajte na nekoj novoj bazi i njenim tablicama. Naša baza `ulazni_racuni` i njene tablice će nam trebati.

Baza se briše naredbom:

```
mysql> drop ulazni_racuni;
```

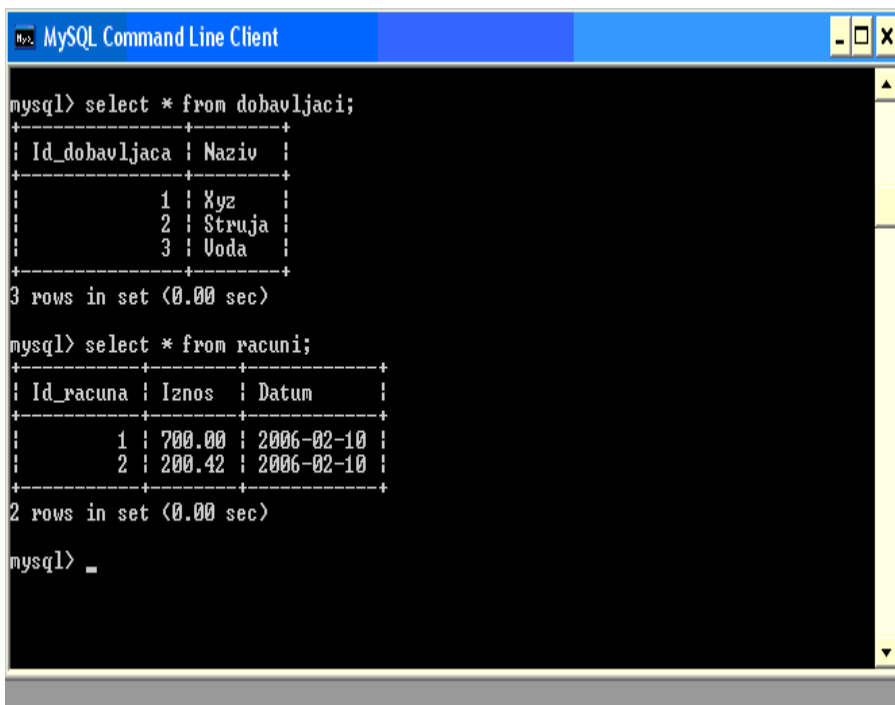
Tablica se briše naredbom:

```
mysql> use ulazni_racuni;  
mysql> drop dobavljac;
```

Sljedi: Promjene u strukturi tablice.

Promjene u strukturi tablice

```
select * from dobavljac; Enter.  
select * from racuni; Enter.
```



```
mysql> select * from dobavljac;
```

Id_dobavljacka	Naziv
1	Xyz
2	Struja
3	Voda

```
3 rows in set (0.00 sec)
```

```
mysql> select * from racuni;
```

Id_racuna	Iznos	Datum
1	700.00	2006-02-10
2	200.42	2006-02-10

```
2 rows in set (0.00 sec)
```

```
mysql> _
```

Primarni ključ

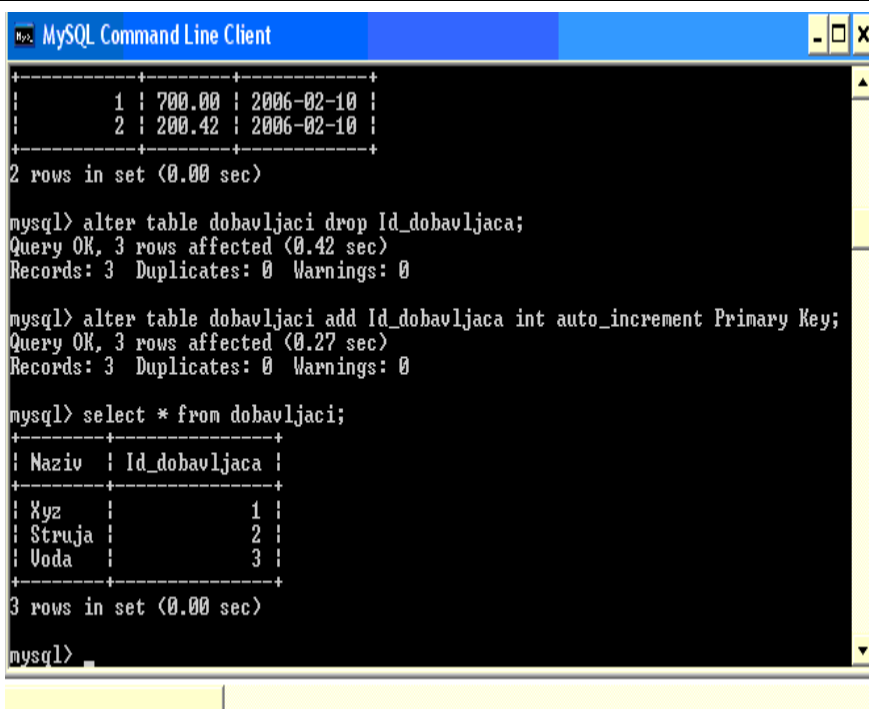
1. Obrisat ćemo polje Id_dobavljacka u tablici dobavljac:

```
mysql> alter table dobavljac drop Id_dobavljacka; Enter.
```

2. Sad ćemo dodati to polje tablici:

```
mysql> alter table dobavljac add Id_dobavljacka int  
auto_increment Primary Key; Enter.
```

```
mysql> select * from dobavljac; Enter.
```



```

mysql>
+-----+-----+-----+
| 1 | 700.00 | 2006-02-10 |
| 2 | 200.42 | 2006-02-10 |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> alter table dobavljac drop Id_dobavljac;
Query OK, 3 rows affected (0.42 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> alter table dobavljac add Id_dobavljac int auto_increment Primary Key;
Query OK, 3 rows affected (0.27 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select * from dobavljac;
+-----+-----+
| Naziv | Id_dobavljac |
+-----+-----+
| Xyz   | 1            |
| Struja | 2            |
| Voda  | 3            |
+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

3. Obrisat ćemo polje Id_racuna u tablici racuni:

```
alter table racuni drop Id_racuna; Enter.
```

4. Sad ćemo dodati to polje tablici:

```
alter table racuni add Id_racuna int auto_increment Primary
Key; Enter.
select * from racuni; Enter.
```

Auto_increment

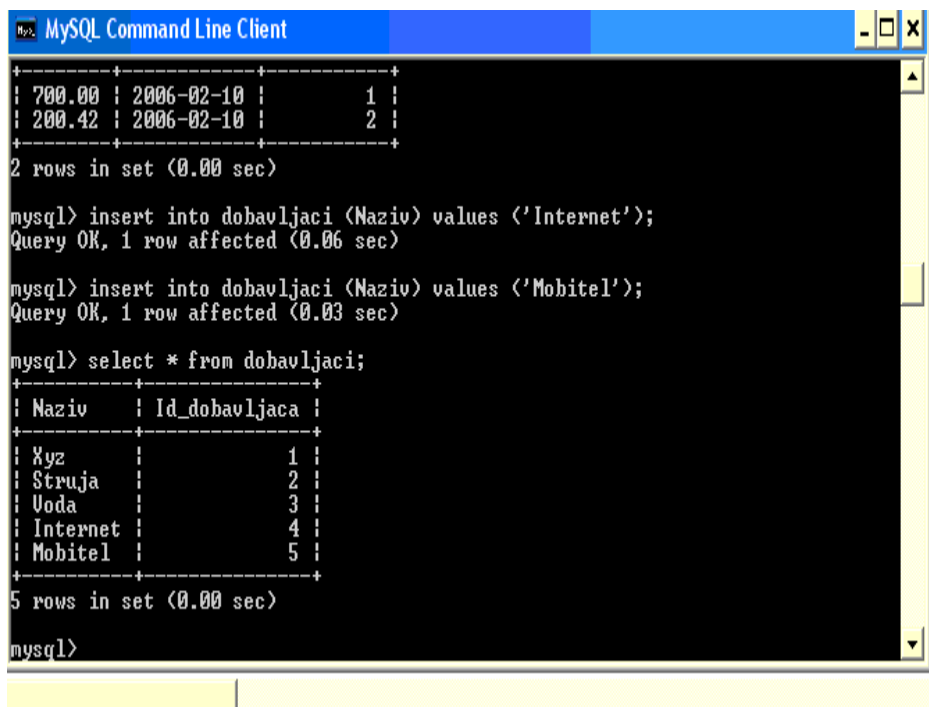
Auto_increment znači da će MySQL umjesto nas upisivati polje Id_racuna, 1, 2, 3, 4... Primary Key znači da je vrijednost u tom polju neponovljiva, jedinstvena. MySQL će brinuti o tome.

1. Sad unosimo podatke u tablicu dobavljac ovako:

```
mysql> insert into dobavljac (Naziv) values ('Internet');
Enter.
mysql> insert into dobavljac (Naziv) values ('Mobitel');
Enter.
```

(Napomena: pritisnite kursorску tipku prema gore i pojavit će se prethodna naredba u kojoj vi samo obrišite Internet i upišite Mobitel).

```
mysql> select * from dobavljac; Enter.
```



```
mysql> insert into dobavljac (Naziv) values ('Internet');
Query OK, 1 row affected (0.06 sec)

mysql> insert into dobavljac (Naziv) values ('Mobitel');
Query OK, 1 row affected (0.03 sec)

mysql> select * from dobavljac;
+-----+-----+
| Naziv | Id_dobavljac |
+-----+-----+
| Xyz   | 1            |
| Struja | 2            |
| Voda  | 3            |
| Internet | 4          |
| Mobitel | 5          |
+-----+-----+
5 rows in set (0.00 sec)

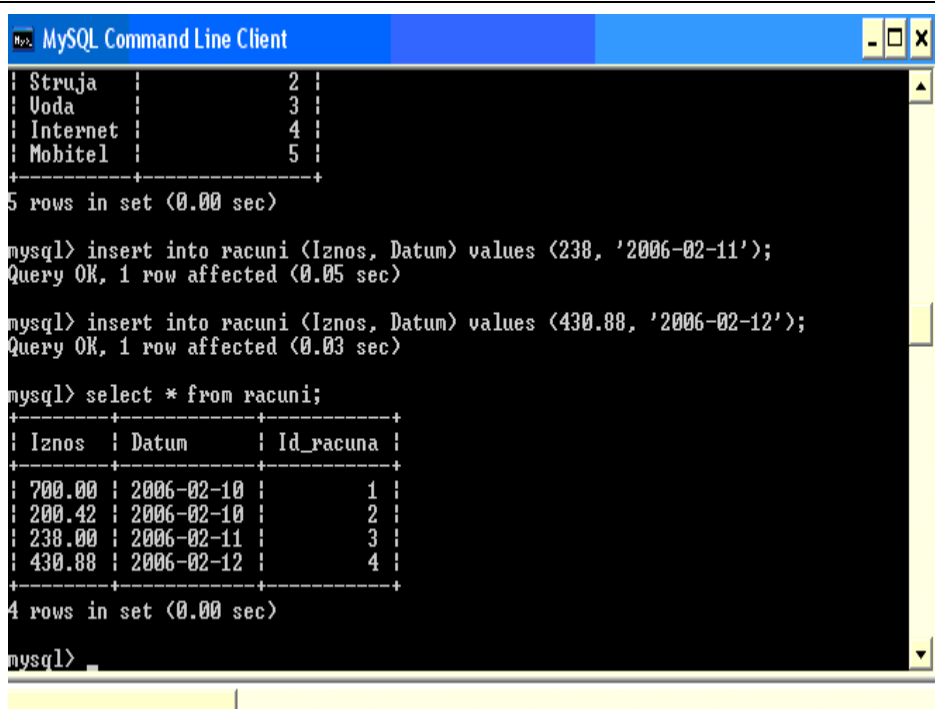
mysql>
```

2. Sad unosimo podatke u tablicu racuni ovako:

```
mysql> insert into racuni (Iznos, Datum) values (238, '2006-02-11'); Enter.
mysql> insert into racuni (Iznos, Datum) values (430.88, '2006-02-12'); Enter.
```

(Napomena: pritisnite kursoršku tipku prema gore i pojavit će se prethodna naredba u kojoj vi samo obrišite prethodne vrijednosti i dodate nove).

```
mysql> select * from racuni; Enter.
```



```

MySQL Command Line Client
+-----+
| Struja |      2 |
| Voda   |      3 |
| Internet |    4 |
| Mobitel |    5 |
+-----+
5 rows in set (0.00 sec)

mysql> insert into racuni (Iznos, Datum) values (238, '2006-02-11');
Query OK, 1 row affected (0.05 sec)

mysql> insert into racuni (Iznos, Datum) values (430.88, '2006-02-12');
Query OK, 1 row affected (0.03 sec)

mysql> select * from racuni;
+-----+
| Iznos | Datum      | Id_racuna |
+-----+
| 700.00 | 2006-02-10 |          1 |
| 200.42 | 2006-02-10 |          2 |
| 238.00 | 2006-02-11 |          3 |
| 430.88 | 2006-02-12 |          4 |
+-----+
4 rows in set (0.00 sec)

mysql>

```

Sljedi: SQL naredba SELECT.

SQL naredba SELECT

```

select Naziv from dobavljac_i order by Naziv; Enter.
select Naziv from dobavljac_i order by Naziv desc limit 3;
Enter.
select Iznos, Datum from racuni order by Datum desc; Enter.
select Iznos, Datum from racuni order by Datum desc, Iznos
asc; Enter.
select Iznos, Datum from racuni order by Datum asc, Iznos
desc; Enter.
select Id_racuna, Iznos, Datum from racuni where Datum =
'2006-02-10' order by Iznos desc; Enter.
select Id_racuna, Iznos, Datum from racuni where Datum =
'2006-02-10' and Iznos = 700.00; Enter.
select Id_racuna, Iznos, Datum from racuni where Datum =
'2006-02-10' and Iznos > 700.00; Enter.
select Id_racuna, Iznos, Datum from racuni where Datum =
'2006-02-10' and Iznos < 700.00; Enter.
select Id_racuna, Iznos, Datum from racuni where Datum =
'2006-02-10' or Iznos > 230.00; Enter.

```

Sljedi: Grupiranje redaka po nekom zajedničkom podatku (Group by)

Grupiranje redaka po nekom zajedničkom podatku (Group by)

```

use ulazni_racuni; Enter.
alter table racuni add Id_dobavljacka int; Enter.
select * from racuni;

```

```

MySQL Command Line Client
mysql> use ulazni_racuni;
Database changed
mysql> alter table racuni add Id_dobavljacka int;
Query OK, 4 rows affected (0.53 sec)
Records: 4 Duplicates: 0 Warnings: 0

mysql> select * from racuni;
+-----+-----+-----+-----+
| Iznos | Datum   | Id_racuna | Id_dobavljacka |
+-----+-----+-----+-----+
| 700.00 | 2006-02-10 | 1 | NULL |
| 200.42 | 2006-02-10 | 2 | NULL |
| 238.00 | 2006-02-11 | 3 | NULL |
| 430.88 | 2006-02-12 | 4 | NULL |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> _

```

```

update racuni set Id_dobavljacka = 2 where Id_racuna = 1;
Enter.
update racuni set Id_dobavljacka = 1 where Id_racuna = 2;
Enter.
update racuni set Id_dobavljacka = 2 where Id_racuna = 3;
Enter.
update racuni set Id_dobavljacka = 1 where Id_racuna = 4;
Enter.
select * from racuni;

```

```

MySQL Command Line Client
mysql> update racuni set Id_dobavljacka = 1 where Id_racuna = 2;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> update racuni set Id_dobavljacka = 2 where Id_racuna = 3;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> update racuni set Id_dobavljacka = 1 where Id_racuna = 4;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from racuni;
+-----+-----+-----+-----+
| Iznos | Datum   | Id_racuna | Id_dobavljacka |
+-----+-----+-----+-----+
| 700.00 | 2006-02-10 | 1 | 2 |
| 200.42 | 2006-02-10 | 2 | 1 |
| 238.00 | 2006-02-11 | 3 | 2 |
| 430.88 | 2006-02-12 | 4 | 1 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

Group by

mysql>select * from racuni group by Id_dobavljacka; Enter.

```

mysql> update racuni set Id_dobavljacka = 1 where Id_racuna = 4;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from racuni;
+-----+-----+-----+-----+
| Iznos | Datum   | Id_racuna | Id_dobavljacka |
+-----+-----+-----+-----+
| 700.00 | 2006-02-10 | 1 | 2 |
| 200.42 | 2006-02-10 | 2 | 1 |
| 238.00 | 2006-02-11 | 3 | 2 |
| 430.88 | 2006-02-12 | 4 | 1 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from racuni group by Id_dobavljacka;
+-----+-----+-----+-----+
| Iznos | Datum   | Id_racuna | Id_dobavljacka |
+-----+-----+-----+-----+
| 200.42 | 2006-02-10 | 2 | 1 |
| 700.00 | 2006-02-10 | 1 | 2 |
+-----+-----+-----+-----+
2 rows in set (0.03 sec)

mysql>
  
```

MySQL je izbacio samo po jedan redak od svakog Id_dobavljacka. Od toga nemamo puno koristi, zato nam trebaju MySQL funkcije.

Sljedi: MySQL funkcije.

MySQL funkcije

```

select Id_racuna, avg(Iznos), Id_dobavljacka from racuni group by
Id_dobavljacka; Enter.
select Id_racuna, sum(Iznos), Id_dobavljacka from racuni group by
Id_dobavljacka; Enter.
select Id_racuna, max(Iznos), Id_dobavljacka from racuni group by
Id_dobavljacka; Enter.
select Id_racuna, min(Iznos), Id_dobavljacka from racuni group by
Id_dobavljacka; Enter.
select count(Id_racuna), Id_dobavljacka from racuni group by Id_dobavljacka;
Enter.
  
```

Zadaci

Zadatak: 1

1. Kreirajte bazu podataka 'skola'.
2. Kreirajte tablicu 'ucenici' koja ima polja Id_ucenika, Ime, Prezime, Razred.

3. Unesite desetak učenika u tablicu 'ucenici'. Neka idu po dvoje-troje u isti razred.
4. Pomoću naredbe 'select' kreirajte upit koji grupira učenike istog razreda.

Test

1. SQL jezik je skriptni jezik.
 - a. Da.
 - b. Ne.
2. U naredbi 'mysql -u root' zaporka je 'root'.
 - a. Da.
 - b. Ne.
3. SQL naredba za kreiranje baze 'ulazni_racuni' je:
 - a. create database ulazni_racuni;
 - b. create ulazni_racuni;
4. SQL naredba za ulaz u bazu 'ulazni_racuni' je:
 - a. drop ulazni_racuni;
 - b. create ulazni_racuni;
 - c. use ulazni_racuni;
5. SQL naredba za kreiranje tablice dobavljacu je:
 - a. use table dobavljacu (Id_dobavljacka int, Naziv char(255));
 - b. drop table dobavljacu (Id_dobavljacka int, Naziv char(255));
 - c. create table dobavljacu (Id_dobavljacka int, Naziv char(255));
6. SQL naredba za unos podataka u tablicu dobavljacu je:
 - a. use into dobavljacu values (1, "Xyz");
 - b. drop into dobavljacu values (1, "Xyz");
 - c. delete into dobavljacu values (1, "Xyz");
 - d. insert into dobavljacu values (1, "Xyz");
7. SQL naredba za upit 'prikaži sve retke i sve stupce tablice dobavljacu' je:
 - a. insert into * dobavljacu;
 - b. group * dobavljacu;
 - c. select * from dobavljacu;
8. SQL naredba za ispravljanje postojećih podataka je:
 - a. select racuni set Datum = "2006-02-10" where Datum = "0000-00-00";

- b. `alter table racuni set Datum = "2006-02-10" where Datum = "0000-00-00";`
 - c. `update racuni set Datum = "2006-02-10" where Datum = "0000-00-00";`
- 9. SQL naredba za brisanje polja u tablici je:
 - a. `update table dobavljac drop Id_dobavljac;`
 - b. `drop table dobavljac drop Id_dobavljac;`
 - c. `alter table dobavljac drop Id_dobavljac;`
- 10. SQL upit za prva tri u listi naziva dobavljača poredanih od ž prema a:
 - a. `update Naziv from dobavljac order by Naziv desc limit 3;`
 - b. `select Naziv from dobavljac order by Naziv asc limit 3;`
 - c. `select Naziv from dobavljac order by Naziv desc limit 3;`

Zaključak

Cilj ove nastavne cjeline je bio upoznati početnike s najčešćim SQL naredbama. Sad kad znamo te SQL naredbe bit će nam vrlo lako pratiti nastavak PHP-a. A u nastavku je riječ o pristupanju MySQL bazi preko PHP stranica koje smo sami napravili.



PHP pristup MySQL bazi

Metodika-----

Više ne moramo kreirati baze i tablice tipkanjem na mysql naredbodavnom retku, već možemo ispisati SQL naredbe u .sql datoteku i pomoću naredbe:

```
mysql -u root < ulazni_racuni.sql
```

kreirati bazu, tablice, popuniti tablice. Tako profesor može unaprijed pripremiti željene baze za vježbanje, kopirati .sql datoteku na mrežu i svaki učenik u trenu kreira tu bazu. I prelazi se na ono što je tema sata, a to je kontakt naših php stranica s bazom podataka.

Ako svaki učenik instalira bazu pomoću .sql datoteke, izbjegavaju se tipografske gresške učenika do kojih bi došlo da sami tipkaju bazu. Izgubilo bi se puno vremena na baze i ostalo bi malo vremena za rad na glavnoj temi: php kodiranje.

Sad ćemo naći razlog našeg vježbanja SQL-a, umjesto korištenja grafičkog sučelja koje bi umjesto nas pisalo SQL.

Niti jedan program nas ne može zamijeniti u pisanju PHP koda. Baza podataka ne razumije PHP, a PHP ne razumije SQL. Kako će onda komunicirati PHP i MySQL? Evo kako:

```
mysql_connect("localhost", "root", "");  
mysql_select_db("ulazni_racuni");  
mysql_query("SELECT * FROM dobavljac");
```

Specijalizirane PHP funkcije u svojim zagradama, kao svoje argumente, prenose MySQL-u potrebne podatke na njemu razumljivom SQL-u. PHP ima tip podataka string. Što znači taj string, to PHP-a ne zanima. Tako taj string može biti bilo koja skupina znakova, pa tako i SQL.

Poslužitelj MySQL će primiti te stringove i razumjeti ih:

- ostvarit će se spoj web stranice s poslužiteljem MySQL
- ostvarit će se spoj s bazom podataka "ulazni_racuni"
- izvršit će se SQL upit "SELECT * FROM dobavljac"

Cilj ove nastavne jedinice je ponavljanje tipkanja ove tri funkcije:

```
mysql_connect("localhost", "root", "");  
mysql_select_db("ulazni_racuni");  
mysql_query("SQL UPIT");
```

dok se ne shvati, uđe pod kožu, proces spajanja php stranice i mysql. To bi trebala postati rutina.

Također, treba raditi primjere sa svim onim SQL upitima iz prethodne nastavne jedinice.

A onda je tu i vježbanje ugradnje podataka u XHTML dokument:

```
mysql_connect("localhost","root","");
mysql_select_db("ulazni_racuni");
$resultat=mysql_query("SELECT * FROM dobavljac");
while($redak = mysql_fetch_array($resultat))
{
    print $redak['ID_dobavljac'] . " " . $redak['Naziv'];
    print "<br />";
}
```

Od funkcije `mysql_fetch_array` nećemo se odvajati do kraja knjige. Ta funkcija iz tipa podatka 'resurs' (`$resultat`) izvuče ASOCIJATIVNI ARRAY:

```
Array ([0] => Array (ID_dobavljac => 1, [Naziv] => Hrana )
      [1] => Array (ID_dobavljac => 2, [Naziv] => Struja )
      [2] => Array (ID_dobavljac => 3, [Naziv] => Voda )
      )
```

iz kojeg možemo vaditi podatke pomoću naziva stupaca u tablici baze podataka.

```
print $redak['ID_dobavljac'] . " " . $redak['Naziv'];
```

Znamo se spojiti na bazu, znamo izvući iz baze RESURS, znamo iz resursa izvući podatke i znamo ih ugraditi u XHTML. To treba dobro izvježbati jer na tom znanju leže sve današnje php web aplikacije.

Udžbenik-----

Napravimo bazu ulazni_racuni

Napomena: datoteka `ulazni_racuni.sql` je u komprimiranoj datoteci koju možete skinuti na adresi www.e92.hr, na mjestu gdje se nudi i ova knjiga.

1. Tipkajte u Notepadu:

```
DROP DATABASE IF EXISTS `ulazni_racuni`;

CREATE DATABASE ulazni_racuni;

USE ulazni_racuni;

DROP TABLE IF EXISTS dobavljac;

CREATE TABLE dobavljac (
    ID_dobavljac int auto_increment,
    Naziv char(255),
    primary key (ID_dobavljac)
) TYPE=MyISAM;

DROP TABLE IF EXISTS racuni;
CREATE TABLE racuni (
```

```
ID_racuna int auto_increment,
Iznos float(7,2),
Datum date,
Id_dobavljacka int,
primary key (ID_racuna)
) TYPE=MyISAM;
```

```
insert into dobavljacki (Naziv) values ("Hrana");
insert into dobavljacki (Naziv) values ("Struja");
insert into dobavljacki (Naziv) values ("Voda");
```

```
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("700", "2006-2-10", "3");
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("200.42", "2006-3-9", "3");
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("10.50", "2006-3-11", "3");
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("18.40", "2006-4-12", "1");
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("22.30", "2006-4-22", "1");
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("18.40", "2006-5-15", "1");
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("18.40", "2006-6-14", "2");
insert into racuni (Iznos, Datum, Id_dobavljacka) values
("18.40", "2006-7-8", "2");
```

2. Pohranite pod imenom `ulazni_racuni.sql` u mapu 'bin'. U mom slučaju je put do te mape ovaj: `c:\program files\mysql\mysql server 4.1\bin`
3. Otvorite Command Prompt (Start/Run/tipkajte cmd, Enter).
4. Dodajte do mape 'bin'. U mom slučaju je to `c:\program files\mysql\mysql server 4.1\bin`.
5. Tipkajte ovu naredbu:

```
mysql -u root < ulazni_racuni.sql
Enter
```

Sad imamo kreiranu bazu `ulazni_racuni`, s unijetim podacima u tablice `racuni` i `dobavljacki`.

6. Tipkajte u Notepad dokumentu i pohranite pod imenom `l.php`:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>

<?php
```

```
mysql_connect("localhost", "root", "");  
mysql_select_db("ulazni_racuni");  
mysql_query("SELECT * FROM dobavljac");  
?>  
  
</body>  
</html>
```

1. Otvorite 1.php u Web pregledniku. Prazna web stranica? Mora biti prazna, jer nigdje nismo tipkali naredbu 'print' ili 'echo', a nema niti HTML-a u sekciji body ili bilo kojeg drugog sadržaja osim php programa (php otoka, kako to ponekad slikovito nazivamo.)

Objašnjenje

Ističem ovo: sav posao oko baza obavljaju php funkcije. Mi samo svaki put iznova u svom kodu koristimo php funkcije! Netko je lijepo rekao: sva snaga PHP-a je u njegovim funkcijama!

PHP kao korisnik

PHP skript na našim web stranicama je korisnik baze kao i svaki drugi korisnik baze. To znači, mora imati 'account' s pravima pristupa poslužitelju MySQL i pravima pristupa dotičnoj bazi podataka. Kad ste instalirali MySQL na svoje računalo pomoću XAMPP-a, automatski je kreiran korisnik 'root' s praznom zaporkom. Taj korisnik root ima pristup poslužitelju MySQL i svim bazama. Ima sva moguća prava. Mi u svojim vježbama koristimo tog korisnika 'root' bez zaporka.

Sjetimo se Command Prompta i MySQL monitora

Sjećate se prethodne nastavne cjeline i kako ste pristupali MySQL bazi.

1. Najprije ste pristupili MySQL poslužitelju, kao korisnik root sa svim mogućim pravima, bez zaporka:

```
mysql -u root  
ili  
mysql -u root@localhost  
Enter
```

2. Onda ste tipkali naredbu za pristup bazi ulazni_racuni:

```
use ulazni_racuni;  
Enter
```

3. I sad ste mogli tipkati upite po volji:

```
select * from dobavljac;  
Enter
```

PHP prolazi isti gore opisani postupak

PHP se spaja na MySQL bazu isto kao što smo se mi spojili pomoću MySQL monitora. Samo što sve to obave php funkcije, a tamo smo to obavljali mi ručno.

Evo tih famoznih php funkcija, poredanih baš kako i treba. Njihovi argumenti im daju potrebne podatke za spoj na bazu.

```
<?php
mysql_connect("localhost", "root", "");
mysql_select_db("ulazni_racuni");
mysql_query("SELECT * FROM dobavljac");
?>
```

Prva funkcija je ostvarila spoj s poslužiteljem MySQL, druga spoj s bazom ulazni_racuni, a treća funkcija je poslala SQL upit bazi.

Ovo je zanimljiv primjer gdje dvoje sugovornika ne razumiju jezik jedan od drugoga, a ipak se odlično sporazumijevaju. Sve baze govore SQL jezikom. PHP govori samo svojim PHP jezikom. Kako su se onda sporazumjeli PHP i MySQL?

Kako se razumiju PHP i MySQL

Trik se krije u onoj trećoj funkciji, mysql_query:

```
mysql_query("SELECT * FROM dobavljac");
```

Ta funkcija je tako skrojena od svojih autora da prenese bazi podataka string, ma kakav on bio i od čega se sastojao. To je kao pošta koja prenosi pakete od vrata do vrata, ali ne zna što je u paketu.

U toj trećoj instrukciji se dogodilo i još nešto. Baza podataka je razumjela upit jer je pisan na njoj razumljivom SQL jeziku:

```
"SELECT * FROM dobavljac"
```

i odmah vratila rezultat nazad php stranici po istoj instrukciji. Tako je u jednoj instrukciji odrađen tour-retour nekog stringa.

Ali zašto je naša stranica prazna?

Gore je rečeno, nema drugih instrukcija koje će je prikazati na web stranici. To su one instrukcije koje počinju s 'print' ili 'echo'. Dodajmo ih, pa da vidimo i rezultat upita.

Vidimo što nam je baza vratila

1. Tipkajte u Notepad:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<?php
mysql_connect("localhost","root","");
```

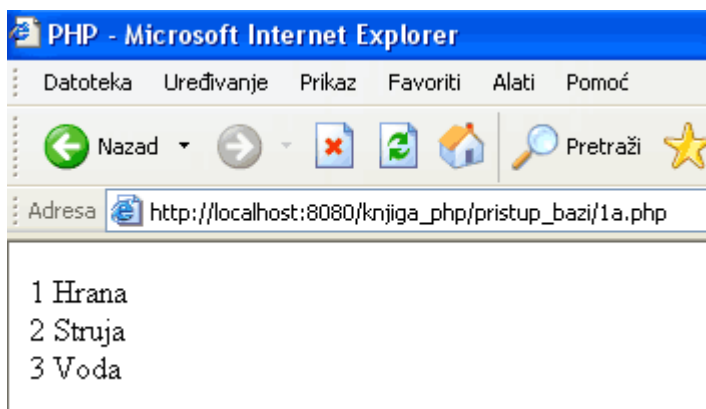


```

mysql_select_db("ulazni_racuni");
$resultat=mysql_query("SELECT * FROM dobavljac");
while($redak = mysql_fetch_array($resultat))
{
    print $redak['ID_dobavljacka'] . " " . $redak['Naziv'];
    print "<br />";
}
?>
</body>
</html>

```

2. Pohranite pod imenom 1a.php.
3. Otvorite 1a.php u pregledniku i vidjet ćete:



Objašnjenje

U ovom kodu je novost:

```

$resultat=mysql_query("SELECT * FROM dobavljac");
while($redak = mysql_fetch_array($resultat))
{
    print $redak['ID_dobavljacka'] . " " . $redak['Naziv'];
    print "<br />";
}

```

Rezultat upita smo pohranili u varijablu \$resultat. U petlji While varijabli dajemo ime 'redak' kako bismo istakli ulogu funkcije mysql_fetch_array(\$resultat). Ta će nas funkcija pratiti kroz cijelu knjigu. Ona u svakom ciklusu petlje While pomakne fokus na sljedeći redak rezultata upita. Svaki redak je jedan Array. Točnije, na engleskom: associative array. To je ovaj array:

```

Array ([0] => Array (ID_dobavljacka => 1, [Naziv] => Hrana )
      [1] => Array (ID_dobavljacka => 2, [Naziv] => Struja )
      [2] => Array (ID_dobavljacka => 3, [Naziv] => Voda )
)

```

Naziv 'associative' ili 'pridružen' dolazi od tud što je svakom podatku pridruženo i ime stupca kojem pripada.

Zahvaljujući tome, mi možemo dohvatiti podatak i uz pomoć naziva stupca, na primjer:

\$redak[1]['Naziv'] će izbaciti 'Struja'. Isto kao da napišemo \$redak[1][1].

A sad želimo vidjeti tablični prikaz

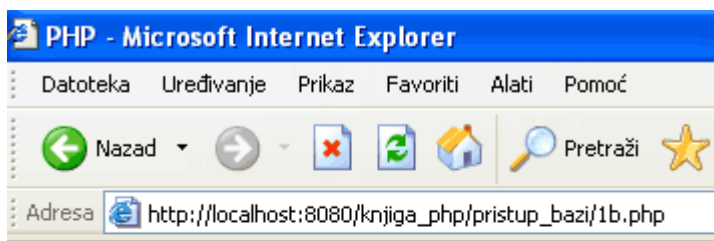
1. U Notepadu napišite ovaj kôd:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<?php
mysql_connect("localhost","root","","");
mysql_select_db("ulazni_racuni");
$rezultat=mysql_query("SELECT * FROM dobavljac");

echo "<table border='1'>
<tr>
<td>ID dobavljača</td>
<td>Naziv</th>
</tr>";

while($redak = mysql_fetch_array($rezultat))
{
    echo "<tr><td>";
    echo $redak['ID_dobavljacka'];
    echo "</td><td>";
    echo $redak['Naziv'];
    echo "</td></tr>";
}
echo "</table>";
?>
</body>
</html>
```

2. Pohranite pod imenom 1b.php. Otvorite u pregledniku i trebali biste vidjeti ovu web stranicu:



ID dobavljača	Naziv
1	Hrana
2	Struja
3	Voda

Zadaci

1. Napišite PHP kôd koji će kreirati bazu 'zadaci'.
2. U bazi 'zadaci' kreirajte tablicu 'zadaci_php'. Tablica ima dva polja: ID, Opis.
3. Unesite podatke u tablicu zadaci_php.
4. Napravite formu za unos podataka u tablicu zadaci_php.
5. Napravite knjigu gostiju.
6. Pogledajte baze koje ste napravili preko grafičkog php sučelja phpMyAdmin.

Rješenja

1. Zadatak:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<?php
mysql_connect("localhost","root","");
mysql_query("create database zadaci");
    print "Baza je kreirana";
?>
</body>
```

```
</html>
```

Provjerite ovako: otvorite Command Prompt, pa otvorite MySQL monitor. Dodite do `mysql>` naredbodavnog retka i tipkajte SQL naredbu:

```
show databases;
```

2. Zadatak

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<?php
mysql_connect("localhost","root@localhost","ivica");
mysql_select_db("zadaci");
mysql_query("create table zadaci_php
  (ID int auto_increment primary key,
   Opis text)");

print "Tablica je napravljena.";

?>
</body>
</html>
```

Provjerite u `mysql>` naredbodavnom retku:

```
use zadaci;
show tables;
```

3. Zadatak

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<?php
mysql_connect("localhost","root@localhost","ivica");
mysql_select_db("zadaci");
mysql_query("insert into zadaci_php (Opis) values
('Napišite php kod za kreiranje tablice')");

print "Podaci su unijeti u tablicu.";

?>
```

```
</body>
</html>
```

Provjerite u mysql> naredbodavnom retku:

```
use zadaci;
select * from zadaci_php;
```

4. Zadatak

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>

<form action="zadatak4.php" method="post">
Opis zadatka: <textarea name="opis"></textarea>
<input type="submit" value="unesi" />
</form>

<?php
mysql_connect("localhost","root@localhost","ivica");
mysql_select_db("zadaci");
mysql_query("insert into zadaci_php (Opis) values
('$ _POST[opis]')");
?>
</body>
</html>
```

Provjerite u mysql> naredbodavnom retku:

```
use zadaci;
select * from zadaci_php;
```

5. Zadatak

Napravite bazu podataka 'kg' i tablicu 'knjiga'. Tablica ima dva polja: 'ime' i 'komentar'. Pomoću MySQL monitora bi to ovako išlo:

```
create database kg; Enter
use kg; Enter
create table (ime char(20) not null, komentar text not null);
Enter
```

Sad napravite ovu php stranicu:

```
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<form action="zadatak5.php" method="post">
```

```

Ime: <input type="text" name="ime" /><br />
Komentar: <textarea name="komentar"></textarea>
<input type="submit" value="unesi" />
</form>
<?php
mysql_connect("localhost","root@localhost","ivica");
mysql_select_db("kg");
mysql_query("insert into knjiga (ime, komentar) values
('$ _POST[ime]', '$ _POST[komentar]')");
$rezultat=mysql_query("SELECT * FROM knjiga");
echo "<table border='1'>
<tr>
<td>Ime</td>
<td>Komentar</th>
</tr>";
while($redak = mysql_fetch_array($rezultat))
{
    echo "<tr><td>";
    echo $redak['ime'];
    echo "</td><td>";
    echo $redak['komentar'];
    echo "</td></tr>";
}
echo "</table>";
?>
</body>
</html>

```

6. U polje address web preglednik tipkajte ovu adresu:

<http://localhost/phpmyadmin/>

Zadaci

1. Zadavajte jedni drugima baze podataka i kreirajte ih.
2. Napravite bazu podataka 'novine' koja ima stupce
 - a. ID
 - b. naslov
 - c. tekst
 - d. datum
3. Koje ćete tipove podataka dodijeliti gornjim stupcima?
4. Pronađite na Webu sve o PHP OPERATORIMA.
5. Pronađite na Webu sve o PHP TIPOVIMA PODATAKA.
6. Kreirajte nekoliko SELECT upita i rezultat prikažite na PHP stranici.
7. Mijenjajte XHTML strukturu u kojoj prikazujete podatke (tablično, divovi, liste itd.)
8. Mijenjajte formatiranje PHP stranica pomoću CSS.

Test

1. DROP DATABASE ... je SQL naredba za _____.
2. DROP TABLE ... je SQL naredba za _____.
3. USE ... je SQL naredba za _____.
4. INSERT ... je SQL naredba za _____.
5. DELETE ... je SQL naredba za _____.
6. UPDATE ... je SQL naredba za _____.
7. SELECT ... je SQL naredba za _____.
8. CREATE ... je SQL naredba za _____.
9. WHERE ... je SQL naredba za _____.
10. LIKE ... je SQL naredba za _____.
11. PHP razumije SQL.
 - a. Da.
 - b. Ne.
12. SQL razumije PHP.
 - a. Da.
 - b. Ne.
13. Svaka FUNKCIJA će se IZVRŠITI samo AKO JE NETKO POZOVE. Taj 'NETKO' može biti instrukcija, događaj, neka vanjska aplikacija ili pak neki 'unutrašnji automatski pozivatelji'. Tko je pozvao ove funkcije:

```
<?php
mysql_connect("localhost", "root", "");
mysql_select_db("ulazni_racuni");
mysql_query("SELECT * FROM dobavljac");
?>
```
14. Da li funkcije zovemo METODE?
 - a. Da.
 - b. Ne.
15. Po čemu na prvi pogled znate da se radi o FUNKCIJI?
 - a. Po okruglim zagradama koje sljede iza imena svake funkcije.
 - b. Po imenu koje ne smije imati razmake.
16. Objasnite ovaj kôd:

```
echo "<tr><td>";
echo $redak['ime'];
echo "</td><td>";
echo $redak['komentar'];
echo "</td></tr>";
```

17. Možemo li pomoću ECHO kreirati po volji veliki XHTML i CSS dokument?



PHP Blog

Metodika-----

Ovo je četvrta nastavna jedinica i u njoj radimo prvu životnu, vrlo popularnu web aplikaciju Blog. Ovo ću vjerojatno ponoviti više puta u knjizi: Svaka web aplikacija uz male ili nikakve izmjene u programskog kodu i veće ili manje izmjene u XHTML-u i CSS-u, postaje nešto drugo. Tako ova aplikacija bez intervencije može biti i Knjiga gostiju, Novine online, Portal, Foto album itd.

Prve tri nastavne jedinice su nas dovele do ove kreativne točke za finalne proizvode. Do sad smo mogli biti kreativni s petljama, s IF strukturama, s varijablama. Tu elementarnu kreativnost sad ćemo primjeniti na korisni, upotrebljivi, tržišno vrijedan konačni proizvod – cjelovita web aplikacija 'Blog'. Konačno ćemo imati svoj Blog!

Svaka web aplikacija ima JAVNI dio i ADMINISTRATIVNI dio. Tako i ova naša. Javni dio su sve web stranice kojima može prići svaki posjetitelj. Administrativni dio čine web stranice kojima mogu prići samo osobe koje uspješno prođu test identifikacije. U našem blogu ćemo identifikaciju provesti preko Forme za identifikaciju.

U ovoj nastavnoj jedinici je obrađen JAVNI dio aplikacije Blog. To je početna stranica, tražilica i forma za dodavanje novog bloga.

Primjenio sam minimalnu moguću strukturu XHTML i minimalno formatiranje CSS. Ukoliko su učenici ranije naučili XHTML i CSS, slobodno im se može pustiti na volju izgradnja i formatiranje web stranica. Ovo je prilika za biti kreativan u XHTML, CSS i PHP.

Za pripreme satova koristio sam Google. Iako nam Internet pruža brdo gotovih primjera, na pripreme sam trošio puno, puno vremena.

Ova knjiga bi bila jako debela i skupa da sam u njoj isjeckao sve gradivo na nastavne cjeline onako kako sam ih izvodio u nastavi.

Pokazao bih u radu na projektoru gotovu aplikaciju Blog i rekao: to je naš konačni cilj. A sad na posao, dio po dio iz sata u sat, tjedna u tjedan.

I onda bih složio pripremu prema satnici:

- Kreiranje baze podataka ili pomoću .sql datoteke koju im podijelim ili rade ručno preko mysql naredbodavnog retka. I unijeli bismo nekoliko podataka, jer radimo najprije početnu stranicu na kojoj se vide blogovi.
- Radimo index.php. To je najjednostavnije: imamo SQL upit SELECT... i prikaz tablice iz baze u našoj XHTML tablici.

- Nakon toga radimo stranicu s Formom i kôd za dodavanje a to je SQL upit s INSERT INTO.

Udžbenik-----

Što ćemo napraviti?

1. Bazu 'blog' koja ima tablicu 'blog'.
2. Početnu stranicu s blogovima.
3. Stranicu za unos blogova.
4. Stranicu za administriranje aplikacije BLOG.
5. Stranicu za logiranje moderatora.
6. Tražilicu

Baza 'blog'

1. Tipkajte u Notepadu:

```
DROP DATABASE IF EXISTS blog1;

CREATE DATABASE blog1;

USE blog1;

DROP TABLE IF EXISTS blog;

CREATE TABLE blog (
  ID int auto_increment primary key,
  subjekt char(30) not null,
  unos text not null,
  datum date,
  autor char(20) not null
) TYPE=MyISAM;
```

1. Pohranite pod imenom blog1.sql u mapu 'bin'. U mom slučaju je put do te mape ovaj: c:\program files\mysql\mysql server 4.1\bin
2. Tipkajte ovu naredbu u Command Prompt ...bin> naredbenom retku:


```
mysql -u root < blog.sql
Enter
```

Napomena: datoteku blog1.sql naći ćete u download kompletu datoteka, kao i sve što sljedi..

Blog - PHP stranice

Sljedeće PHP stranice tipkajte u Notepadu.

index.php

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<h1>Blog</h1>
<a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<?php
    mysql_connect("localhost", "root", "");
    mysql_select_db("blog1");
    $rezultat = mysql_query("SELECT subjekt, unos, datum,
autor FROM blog order by ID desc;");
    echo "<table border='1'>
<tr valign='top'>
<td>Subjekt</td>
<td>Unos</td>
<td>Datum</td>
<td>Autor</td>
</tr>";
    while($redak = mysql_fetch_array($rezultat))
    {
        echo "<tr valign='top'><td>";
        echo $redak['subjekt'];
        echo "</td><td>";
        echo $redak['unos'];
        echo "</td><td>";
        echo $redak['datum'];
        echo "</td><td>";
        echo $redak['autor'];
        echo "</td></tr>";
    }
    echo "</table>";
?>
</body>
</html>

```

dodaj.php

```

<a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<form action="dodaj_pohrani.php" method="GET">
Subjekt: <input type="text" name="subjekt" /><br />
Unos: <textarea name="unos"></textarea><br />
Autor: <input type="text" name="autor" /><br />
<input type="submit" />
</form>

```

dodaj_pohrani.php

```

<?php

```

```

mysql_connect("localhost", "root", "");
mysql_select_db("blog1");

$result = mysql_query("INSERT INTO blog (subjekt, unos,
datum, autor) values
('$_GET[subjekt]', '$_GET[unos]', now(), '$_GET[autor]')");

if (mysql_affected_rows() == 1) {
?>
Uspješno unešeno! <a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<?php
}
else
{
?>
Nije unijeto! <a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<?php
}
?>

```

admin.php

```

<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<h1>Administracija bloga</h1>
<a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<?php
    mysql_connect("localhost", "root", "");
    mysql_select_db("blog1");
    $rezultat = mysql_query("SELECT ID, subjekt, unos, datum,
autor FROM blog order by ID desc;");

echo "<table border='1'>
<tr valign='top'>
<td>Subjekt</td>
<td>Unos</td>
<td>Datum</td>
<td>Autor</td>
<td>&nbsp;</td>

```

```

<td>&nbsp;</td>
</tr>";
while($redak = mysql_fetch_array($rezultat))
{
    echo "<tr valign='top'><td>";
    echo $redak['subjekt'];
    echo "</td><td>";
    echo $redak['unos'];
    echo "</td><td>";
    echo $redak['datum'];
    echo "</td><td>";
    echo $redak['autor'];
    echo "</td><td>";
    $id = $redak['ID'];
    echo "<a href='delete.php?ID=$id'>Obriši</a>";
    echo "</td><td>";
    echo "<a href='edit.php?ID=$id'>Promijeni</a>";
    echo "</td><td>";
}
echo "</table>";
?>

</body>
</html>

```

delete.php

```

<?php
    mysql_connect("localhost", " root", "");
    mysql_select_db("blog1");
    $rezultat = mysql_query("DELETE FROM blog WHERE
ID={$_GET['ID']} LIMIT 1");

    if (mysql_affected_rows() == 1)
    {
        print "Promjene su unijete u tablicu.<br />";
        print "<a href='index.php'>Početna stranica</a><br />";
        print "<a href='admin.php'>Admin</a><br />";
        print "<a href='dodaj.php'>Dodaj</a><br />";
    }
    else
    {
        print "Niste ništa mijenjali.<br />";
        print "<a href='index.php'>Početna stranica</a><br />";
        print "<a href='admin.php'>Admin</a><br />";
        print "<a href='dodaj.php'>Dodaj</a><br />";
    }
?>

```

edit.php

```

<?php
    mysql_connect("localhost", " root", "");
    mysql_select_db("blog1");
    $rezultat = mysql_query("SELECT * FROM blog where
ID={$_GET['ID']} LIMIT 1");

```

```

print "<form action='edit_pohrani.php' method='POST'>";
print "<table align='center' border='0' width='80%'>";

    while ($redak = mysql_fetch_assoc($rezultat)) {
        extract($redak, EXTR_PREFIX_ALL, "kartelo");

print "<tr valign='top'>";
    print "<td>&nbsp;  </td>";
    print "<td>Subjekt</td>";
    print "<td>Autor</td>";
    print "<td>Unos</td>";

print "<tr valign='top'>";
    print "<td><input type='hidden' name='id'
value='$kartelo_ID'></td>";
    print "<td><input type='text' name='subjekt'
value='$kartelo_subjekt'></td>";
    print "<td><input type='text' name='autor'
value='$kartelo_autor'></td>";
    print "<td><input name='unos' value=$kartelo_unos
/></td>";
    print "<td><input type='submit' name='submit'
value='Ažuriraj'></td>";
print "</form>";
    print "<td>
<form action='admin.php' method='post'>
<input type='submit' name='submit' value='Odustani'>
</form></td>";
print "</tr>";
    }
print "</table>";
?>

```

edit_pohrani.php

```

<?php
    mysql_connect("localhost", "localhost@root", "ivica");
    mysql_select_db("blog1");
    $result = mysql_query("UPDATE blog SET
subjekt='$_POST[subjekt]', unos='$_POST[unos]',
autor='$_POST[autor]' where ID='$_POST[id]'");

    if (mysql_affected_rows() == 1)
    {
        print "Promjene su unijete u tablicu.";
        print "<td>
<form action='admin.php' method='post'>
<input type='submit' name='submit' value='Admin'>
</form></td>";
    }
    else
    {
        print "Niste ništa mijenjali.";
    }

```

```

        print "<td>
        <form action='admin.php' method='post'>
        <input type='submit' name='submit' value='Admin'>
        </form></td>";
    }
    ?>

```

Objašnjenje

index.php

1. U polje address web preglednika sam tipkao ovaj url:
http://localhost:8080/knjiga_php/blog/index.php
 Enter
2. Web poslužitelj Apache je primio zahtjev, pronašao stranicu index.php i predao je u 'ruke' php engine-u ili php interpreteru, jer stranica ima nastavak.php. Kad stranica ima nastavak .html, Apache je vraća klijentu takvu kakva je.
3. I tako je php interpreter započeo s procesuiranjem index.php. To znači da je index.php učitana u RAM servera i da procesor uzima redak po redak i uz pomoć php interpretera procesuirao php kôd.
4. Sav HTML skript preskače, ne procesuirao, već ga ostavlja takvog kakav je. Procesuiranje počinje od početnog php graničnika <?php i traje do krajnjeg php graničnika ?>.
5. Prve dvije instrukcije spajaju naš php program s bazom podataka:

```

mysql_connect("localhost", "localhost@root", "ivica");
mysql_select_db("blog1");

```

6. Sljedeća instrukcija u kojoj je SQL upit:

```

$resultat = mysql_query("SELECT subjekt, unos, datum, autor
FROM blog order by ID desc;");

```

Nakon izvršenja instrukcije s ovim upitom, u RAM-u poslužitelja se nalazi cijela tablica 'blog'.

7. Pomoću petlje While i funkcije mysql_fetch_array, iz te tablice 'blog' izvlačimo podatke koji nam trebaju i gdje nam trebaju. Funkcija mysql_fetch_array uzme iz memorije RAM-a (iz varijable \$resultat) prvi redak podataka i napravi od njih 'association' array, na primjer:

```

subjekt=>"Neki subjekt"
unos=>"Neki unos"
datum=>"Neki datum"
autor=>"Neki autor"

```

Mi tada pristupamo tom arrayu u RAM-u i uzimamo podatke koji nam trebaju i gdje nam trebaju. Podatke dohvaćamo preko pridruženog (asocijativnog) imena stupca, subjekt, unos, datum, ili autor:

```

$redak['subjekt']

```



```

while($redak = mysql_fetch_array($rezultat))
{
    echo "<tr valign='top'><td>";
    echo $redak['subjekt'];
    echo "</td><td>";
    echo $redak['unos'];
    echo "</td><td>";
    echo $redak['datum'];
    echo "</td><td>";
    echo $redak['autor'];
    echo "</td></tr>";
}

```

8. Ostali kôd je samo gradio HTML konstrukciju pomoću naredbe 'echo'. Kad je proces došao do krajnjeg graničnika `?>`, procesor je završio procesuiranje i obrisao sve tragove u RAM-u:

- obrisao je spoj na poslužitelj MySQL
- obrisao je spoj na bazu blog1
- obrisao je tablicu 'blog'
- obrisao je array nastao u posljednjem krugu petlje (to je obrisano već nakon zadnje vitičaste zagrade u tijelu petlje)
- obrisao je sve globalne varijable koje nastanu automatski sa svakim zahtijevom i koje žive u RAM-u servera od `<html>` do `</html>`, tj. dok god traje izgradnja web stranice u tom RAM-u.

Web stranica je složena i Apache je vraća nama. Ujedno briše tu web stranicu iz RAM-a servera i gubi se svaki trag o nama i našem zahtjevu te stranice. Ako kliknemo na Refresh gumb ili F5 i ponovo zatražimo istu stranicu, cijeli postupak se ponavlja, a poslužitelj prihvaća naš zahtjev kao da mu je to prvi put.

Tako smo dobili u svom klijent programu stranicu `index.php`, čiji izvorni kôd izgleda ovako:

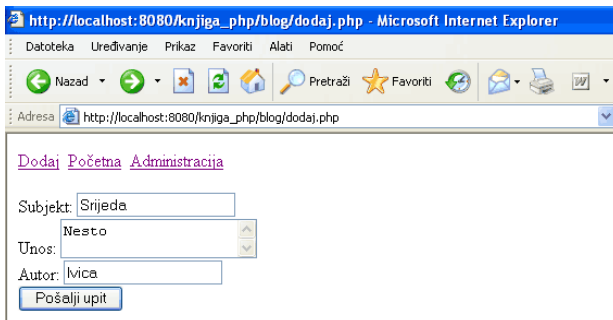
```

<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<h1>Blog</h1>
<a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<table border='1'>
<tr valign='top'>
<td>Subjekt</td>
<td>Unos</td>
<td>Datum</td>
<td>Autor</td>

```


argumente, kao što su type i name. U našoj priči je momentalno važan ovaj argument 'name'.

6. Tipkajmo ove podatke u formu:



Sve što tipkate u elemente `<input>`, dospijeva u RAM vašeg računala pod ovakve adrese:

`name=subjekt=>value=Srijeda`

`name=unos=>value=Nešto`

`name=autor=>value=Ivica`

7. Odaberimo gumb 'Pošalji upit' (Kod mene je lokalizirana verzija Windows XP pa na gumbu submit piše Pošalji upit).

Tako smo aktivirali metodu GET koja kreira ovakav URL, na osnovi onog što je pronašla u memoriji našeg računala:

`http://localhost:8080/knjiga_php/blog/dodaj_pohrani.php?subjekt=Srijeda&unos=Nesto&autor=Ivica`

Svaki element `<forma>` ima i argument 'action'. Argument 'action' ima za vrijednost adresu web stranice koja će biti pozvana kad kliknemo na input tip submit, kojeg također ima svaka forma. U našem slučaju je to stranica `dodaj_pohrani.php`.

Što metoda GET (ili metoda POST) čini? Nakon što smo učinili klik na submit, metoda GET, onom što piše pod 'action'

`dodaj_pohrani.php`

`doda`

`?`

`još doda`

`subjekt=Srijeda`

`pa doda`

`&`

`pa doda`

`unos=Nesto`

`pa doda`

`&`

pa doda

```
autor=Ivica
```

Dakle dodaje parove name=value. Prvi par dodaje nakon znaka upitnik ?, a sve ostale nakon znaka and &.

U nastavku čitajte što slijedi nakon što Apache zaprimi ovaj url:

```
http://localhost:8080/knjiga_php/blog/dodaj_pohrani.php?subjekt=Srijeda&unos=Nesto&autor=Ivica
```

dodaj_pohrani.php

1. Link:

```
http://localhost:8080/knjiga_php/blog/dodaj_pohrani.php?subjekt=Srijeda&unos=Nesto&autor=Ivica
```

poziva stranicu dodaj_pohrani.php i RAM-u servera kreira globalne varijable:

```
$_GET[subjekt]="Srijeda"
```

```
$_GET[unos]="Nesto"
```

```
$_GET[autor]="Ivica"
```

Globalne varijable su u RAM-u na raspolaganju su PHP kôdu.

2. PHP kôd najprije kreira vezu s poslužiteljem MySQL, pa vezu s bazom 'blog1' i onda šalje SQL upit toj bazi:

```
$result = mysql_query("INSERT INTO blog (subjekt, unos, datum, autor) values ('$_GET[subjekt]', '$_GET[unos]', now(), '$_GET[autor]')");
```

SQL upit "INSERT INTO..." kreirao je u tablici 'blog' novi redak i pohranio vrijednosti varijabli:

```
$_GET[subjekt]="Srijeda"
```

```
$_GET[unos]="Nesto"
```

```
$_GET[autor]="Ivica"
```

3. Za vrijednost polja Datum u tom retku, pohranio je iznos kojeg izbaci funkcija NOW(). Ta funkcija i u Accessu i u MySQL-u izbacuje datum tekući datum našeg računala.
4. Slijedi if konstrukcija. Funkcija mysql_affected_rows() uvijek poprimi vrijednost 1 ako se dogodi uspješna kreacija novog retka, pa je ta funkcija pogodna za povratnu informaciju o uspješnom ili neuspješnom unosu u bazu:

```
if (mysql_affected_rows() == 1) {
?>
Uspješno unešeno! <a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<?php
}
```

```

else
{
?>
Nije unijeto! <a href="dodaj.php">Dodaj</a>&nbsp;
<a href="index.php">Početna</a>&nbsp;
<a href="admin.php">Administracija</a>
<?php
}
?>

```

Napomena: u ovom kôdu je posebno zanimljivo kako HTML možemo pisati i bez ključne riječi echo. Naime, mi zatvorimo php kôd s `?>` ispišemo potrebni HTML, pa opet otvorimo `<?php` kad nam treba pa opet zatvorimo `?>` kad nam ne treba.

Pravimo puno malih php otoka u HTML dokumentu.

GET i POST

Metode GET i POST rade isti posao, a razlikuju se u sljedećem:

1. GET prenosi url string preko polja Address pa je taj string svima vidljiv.
2. GET je ograničen samo na 256 znakova.
3. POST nema ograničenja broja znakova i ne prenosi se preko polja address, pa je tako mnogo sigurniji. To su dovoljni razlozi da uvijek kad je forma u pitanju koristimo POST.
4. GET je izvrsna metoda kad želimo linku nešto dodati, pa na osnovu tog dodatka na serveru graditi IF strukture, što ćemo imati priliku vidjeti u sljedećem primjeru stranice admin.php:

```

echo "<a href='delete.php?ID=$id'>Obriši</a>";
echo "<a href='edit.php?ID=$id'>Promijeni</a>";

```

Nastavak .php ili .html

Na stranici index.php smo koristili SELECT za učitavanje cijele tablice 'blog' u web stranicu. Na stranici dodaj.php smo koristili formu za dodavanje. Ova stranica na sebi nema php kôda pa smo joj mogli dati i nastavak .htm ili .html. Sve bi isto radilo. Ali kad već radite u php, dobra praksa je svim stranicama dati nastavak .php, jer nikad ne znate kad ćete kojoj stranici ipak dodati i php kôd. 'Zastoj' koji radi toga trpimo jer stranica nepotrebno mora proći kroz php interpreter je zanemariv.

admin.php

Niti jedna aplikacija ne može se osloniti samo na SQL naredbe SELECT i INSERT. Prva učitava cijele tablice iz baze na naše web stranice, a druga preko web stranica puni te tablice u bazi.

Želite promijeniti nešto u tekstu koji ste poslali u tablice ili je tamo već od ranije. Kako ćete unijeti u tablice te promjene? Zato postoji SQL naredba UPDATE. A ako hoćete cijeli redak u tablici obrisati, za to postoji SQL naredba DELETE.

Stranica admin.php upravo za to služi.

Prva i druga naredba su nas spojili s bazom. Slijedi ovaj upit:

```
$rezultat = mysql_query("SELECT ID, subjekt, unos, datum,
autor FROM blog order by ID desc;");
```

Opet upit SELECT i to potpuno isti kao na stranici index.php. Tako je, jer trebamo vidjeti cijelu tablicu 'blog', kojoj ćemo onda samo dodati linkove 'Promijeni' i 'Obriši'. Idemo odmah na te linkove jer su oni jedino po čemu se ova stranica razlikuje od index.php:

```
$id = $redak['ID'];
echo "<a href='delete.php?ID=$id'>Obriši</a>";
echo "<a href='edit.php?ID=$id'>Promijeni</a>";
```

U svakom prolazu petlje While, pohranio sam \$redak['ID'] u varijablu \$id. Tu pohranjenu vrijednost koristim za GET konstrukciju o kojoj sam pričao ranije:

```
echo "<a href='delete.php?ID=$id'>Obriši</a>";
echo "<a href='edit.php?ID=$id'>Promijeni</a>";
```

Link Obriši poziva stranicu delete.php. Međutim, stranica delete.php ne zna koji redak u tablici ja želim obrisati. Zato ja imam pohranjene ID-ove od svakog retka i ja ih upisujem iza delete.php, tako što dodajem ? i pišem ID=\$id

delete.php?ID=\$id

ili

edit.php?ID=\$id kad želim vršiti promjene u tom retku. Umjesto ?ID mogao sam pisati ?bilo što, ali uvijek biramo imena koja nam i govore o čemu se radi.

U stranici admin.php je ovo bila jedina novost. Puno se zanimljivije stvari događaju na stranicama delete.php i edit.php, pa pogledajmo.

edit.php

U prva dva retka se spajamo na bazu. Sljedi SQL upit:

```
$rezultat = mysql_query("SELECT * FROM blog where
ID={$_GET['ID']} LIMIT 1");
```

Što hvata ovaj upit? Hvata ID retka na kojeg smo u prethodnoj stranici kliknuli. Obratite pozornost na sintaksu:

```
ID={$_GET['ID']}
```

Evo mjesta gdje nam je GET metoda najelegantnije rješenje za prijenos podatka 'na koji ID retka smo kliknuli', sa stranice admin.php na stranicu edit.php.

Ostalo je kreiranje forme i INPUT elemenata, kroz što smo već prošli. Ovdje međutim ima jedna novost.

Upotrijebio sam funkciju extract(\$redak, EXTR_PREFIX_ALL, "kartelo"); u tijelu petlje While:

```
while ($redak = mysql_fetch_assoc($rezultat)) {
    extract($redak, EXTR_PREFIX_ALL, "kartelo");
```

Funkcija `mysql_fetch_assoc`, kao što znamo, kreira asocijativne arraye. Funkcija `EXTRACT()` samo još dodaje neki string, u mom primjeru je to 'kartelo' (samo da pokažem da možete pisati što želite) koji će učiniti sljedeće. Ne morate više izvlačiti podatke iz asocijativnog arraya ovako:

```
$redak['autor']
```

nego to možete učiniti ovako:

```
$kartelo_autor
```

U čemu je razlika?

Ova nova varijanta nema na primjer niti kutne zagrade niti navodnike, što i te kako može biti prednost u kôdu punom navodnika.

Tako mi u ovoj stranici super radi kad napišem:

```
print "<td><input type='text' name='autor'
value='$kartelo_autor'></td>";
```

a ne radi mi kad napišem:

```
print "<td><input type='text' name='autor'
value=$redak['autor']
></td>";
```

Unutar navodnika " " možete samo pisati jednostruke navodnike '. Naredba `PRINT` traži pisanje unutar navodnika " ". Sve unutar mora imati, ako mora, samo '. Kako riješiti problem kad i unutar navodnika " " treba još dvije razine navodnika kao na primjer u ovom slučaju:

```
print "..... value=\"$redak['autor']\"....."
```

Dobro, to smo riješili s funkcijom `extract`.

Na ovoj stranici nema ništa više zanimljivo. To ste već dosta sretali, ali još jednom obratite pozornost na činjenicu da je ovdje cijeli HTML kojeg ćete ugledati na svom klijent programu, napravljen u `RUNTIME-u`, tj. napravio ga je PHP tijekom procesuiranja. Tu je glavnu ulogu odigrala naredba `PRINT`, a mogla je to biti i naredba `ECHO`.

Pogledajmo još samo argument `ACTION` pa ćemo saznati kud nas put dalje vodi:

```
print "<form action='edit_pohrani.php' method='POST'>";
```

Vodi nas na stranicu `edit_pohrani.php`. Kao što nas je stranica `dodaj.php` vodila na stranicu `dodaj_pohrani.php`, tako nas i `edit.php` vodi na stranicu `edit_pohrani.php`. Jedne imaju forme kao grafičko sučelje prema čovjeku. Druge imaju PHP kôd koji će napraviti sve ono 'iza scene': prenijeti podatke iz forme u bazu podataka.

edit_pohrani.php

Cijeli posao oko unosa promjena u određeni redak tablice, obavila je ova instrukcija:

```
$result = mysql_query("UPDATE blog SET
subjekt='$_POST[subjekt]', unos='$_POST[unos]',
autor='$_POST[autor]' where ID='$_POST[id]'");
```

Nama najzanimljiviji u instrukciji je SQL upit. Vidimo da je ovdje SQL naredba UPDATE glavna. Zašto nismo mogli vratiti podatke u tablicu s INSERT? Nismo zato što INSERT uvijek kreira novi redak u tablici. UPDATE je ta naredba koja vraća redak tamo gdje je i bio i u njega stavlja promijenjene podatke.

U funkcijama `mysql_query()` mi pokazujemo svoje znanje iz SQL-a. Nema još danas automatike koja će taj SQL umjesto vas tipkati. Zato, slušajte svog učitelja ako vam kaže "Ne koristite phpMyAdmin, grafičko sučelje za upravljanje MySQL bazama. Koristite MySQL monitor."

U grafičkom sučelju kreirate bazu, tablice, unosite podatke, brišete, mijenjate, sve s klik, klik. Ne tipkate niti jedan redak SQL-a. Vi taj SQL možete vidjeti, ali ako ga vi i ne tipkate nećete ga nikad naučiti.

U MySQL monitoru vi tipkate čisti SQL i tako ga učite na najizravniji mogući način. Kad kasnije počnete tipkati taj isti SQL u svoje PHP stranice, bit ćete zadovoljni sobom.

U sintaksi:

```
$_POST[subjekt]
```

'subjekt' dolazi od vrijednosti argumenta 'name' kao npr. `<INPUT name='subjekt'....>`, a `$_POST` je objekt koji prihvaća toliko globalnih varijabli tipa `$_POST[name iz nekog inputa]` koliko FORMA ima elemenata INPUT.

Globalne varijable, kao i lokalne, kao i cijeli RUNTIME, cijeli svoj život provedu samo na server strani, u RAM-u servera. U našem slučaju je to Apache na našem računalu čija je adresa LOCALHOST ili IP adresa 127.0.0.1

delete.php

Ovaj kôd na stranici admin.php:

```
echo "<a href='delete.php?ID=$id'>Obriši</a>";
```

poziva GET metodu koja preko address polja pošalje serveru url:

```
http://.....delete.php?ID=4
```

Tako stranica delete.php dospije u RUNTIME u kojem se najprije spoji na bazu, a onda SQL naredbom DELETE obriše redak u tablici čiji je ID=4.

```
$rezultat = mysql_query("DELETE FROM blog WHERE  
ID={$_GET['ID']} LIMIT 1");
```

Ovaj LIMIT 1 tek toliko ograničava ovaj upit još dodatno na samo jedan redak. Može i bez tog LIMIT 1. Zašto? Pročitajte u nastavku.

I ovdje smo kao kod naredbe unosa INSERT, koristili funkciju:

```
if (mysql_affected_rows() == 1)
```

za provjeru je li brisanje retka uspjelo. Funkcija `mysql_affected_rows()` poprima vrijednost 1, ne samo kad se dogodi promjena UPDATE, ili unos INSERT novog retka, već i kad nestane DELETE nekog retka.

Auto_increment primary key index

ID u tablici blog smo definirali kao auto_increment i kao primary key.

```
CREATE TABLE blog (  
  ID int auto_increment primary key,  
  subjekt char(30) not null,  
  unos text not null,  
  datum date,  
  autor char(20) not null  
) TYPE=MyISAM;
```

Auto_increment znači da će PHP automatski umjesto nas svakom novom retku u tablici blog, dodijeliti ID=redni broj od 1.... dalje.

Uz to svojstvo gotovo obvezno ide i primary key. Primary key znači da je polje ID ne ponovljivo. Kao takvo, polje ID jednoznačno označuje redak u tablici i jedini je na koga se možemo osloniti da će nepogrešivo odrediti redak. Zato ID koristimo u svim našim kodiranjima za brisanje retka, ediciju i sl.

Ako obrišete na primjer redak ID=4, vaša tablica će imati retke ID=1, 2, 3, 5...

Vi ne možete ubaciti naknadno novi redak ID=4. Jer INSERT samo unosi nove retke na dnu tablice s novim rednim brojem u polju ID. Postojeći brojevi ID su zauvijek iskorišteni. Zato, dobro razmislite prije nego obrišete redak. Sjetite se da postoji i UPDATE i da ga u cijelosti možete promijeniti, pa i nema potrebe za DELETE i INSERT i kad bi se moglo tako.

Svojstva auto_increment i primary key automatski grade i INDEX nad stupcem ID. Index je jedna uobičajena funkcija MySQL poslužitelja, kao i svih drugih RDBMS (Relation Database Management System) - sustava za upravljanje bazama podataka.

Index čini nad nekim poljem tablice ono što ljudi čine telefonskom imeniku kad poredaju prezimena po abecedi, a za svako prezime, poredaju imena po abecedi. I tako se mi u tim knjigama možemo brzo snaći.

Istu stvar index čini polju u tablici. I onda je pretraživanje po tom polju brže. S indexima ćete se još sresti kad bude riječi o tražilicama.

Zadaci

1. Promijenite strukturu stranica pomoću XHTML.
2. Formatirajte stranice pomoću CSS.
3. Prepravite aplikaciju BLOG u aplikaciju KNJIGA GOSTIJU.
4. Prepravite aplikaciju BLOG u CMS-NOVINE aplikaciju. CMS je skraćenica od Content Managment System. To je općenit naziv za sve stranice na kojima se 'upravlja sadržajem'. Zato slobodno možemo reći da je i BLOG jedna CMS aplikacija, kao i sve koje vidimo na Webu. Samo o našoj kreativnosti ovisi što će posjetitelj vidjeti na ekranu i kako će doživjeti GRAFIČKO i FUNKCIONALNO sučelje naše php aplikacije.

5. Prepravite BLOG php kôd, XHTML strukturu i CSS u PORTAL SVOJE ŠKOLE, svoj OSOBNI PORTAL ili PORTAL SVOJE FIRME.
6. Dodajte TRAŽILICU na početnu stranicu aplikacije BLOG.

Rješenje:

Kopirajte stranicu index.php u trazilica.php

Instrukciju \$rezultat... zamjenite s ovom instrukcijom:

```
$rezultat = mysql_query("SELECT subjekt, unos, datum, autor
FROM blog WHERE unos LIKE '%$_POST[trazi]%' order by ID
desc;");
```

Dodajte ovu formu na stranicu index.php

```
<form action="trazilica.php" method="post">
<input type="text" name="trazi" />
<input type="submit" value="Traži" />
</form>
```

Napomena: Ovdje su ključni SQL izrazi WHERE i LIKE. Kasnije u knjizi, u aplikaciji CMS je primijenjena tražilica korištenjem MySQL indexa.

Test

1. Koja je ključna riječ u SQL upitu za UNOS podataka u bazu?
2. Koja je ključna riječ u SQL upitu za BRISANJE retka u bazi?
3. Koja je ključna riječ u SQL upitu za IZMJENE postojećih podataka u bazi?
4. Koja je ključna riječ u SQL upitu za PRIKAZ podataka na našoj php stranici?
5. Pomoću koje funkcije se PHP spoji na MySQL poslužitelj?
6. Koja su tri osnovna argumenta funkcije za spajanje na MySQL poslužitelj?
7. Pomoću koje funkcije se PHP spoji na bazu podataka, i što je argument toj funkciji?
8. Objasni ulogu WHILE petlje.
9. Objasni ulogu funkcije mysql_fetch_array().
10. Što je to asocijativni array?
11. Da li smo u ovim instrukcijama varijabli ID mogli dati bilo koje ime, a ne ID?

```
echo "<a href='delete.php?ID=$id'>Obriši</a>";
echo "<a href='edit.php?ID=$id'>Promijeni</a>";
```

12. Što se događa nakon što kliknemo na link Obriši u ovom kôdu:


```
echo "<a href='delete.php?ID=$id'>Obriši</a>";
```
13. Što se događa nakon što kliknemo na link Promijeni u ovom kôdu:


```
echo "<a href='edit.php?ID=$id'>Promijeni</a>";
```




PHP Login

Metodika-----

Ovdje radimo kôd za logiranje, identifikaciju osobe koja ima pravo pristupa administratorskim stranicama. Identifikacija se obavlja preko forme na kojoj upisujemo korisničko ime i zaporku.

Naša aplikacija funkcionira ovako. Svatko ima pravo anonimno, pod nadimkom, objavljivati u Blog što želi. Vlasnik Bloga, osigurava osobu koja će biti moderator Bloga. Zadatak moderatora je da prati sadržaj Bloga i ukloni onaj za kojeg smatra da je nodoličan.

Moderator koristi više stranica: na jednoj stranici se logira, na drugoj intervenira u sadržaj, na trećoj briše, četvrtoj editira itd. Zahvaljujući SESIJI ne mora prolaziti postupak identifikacije za svaku stranicu posebno.

Sesiju sam detaljno objasnio u nastavku, pa ovdje ne bih imao ništa reći. Dobro je napraviti i nekoliko vježbi na temu sesija.

Udžbenik-----

Što ćemo naučiti?

Glavna tema ovdje je sesija (session). To ponavljam više puta jer smatram jako važnim: klijent i poslužitelj znaju sve jedan o drugom samo dok su u vezi. Čim je poslužitelj poslao web stranicu prema klijentu, automatski je obrisao sve varijable i sve moguće tragove u svojoj memoriji o toj maloprijašnjoj vezi. Vaš svaki novi klik na link je za poslužitelj potpuno novi zahtjev i on nema pojma da ste maloprije bili u vezi.

Ovdje je riječ o sigurnosti. Postoji jedna ili više stranica kojoj smijete prići samo vi. Vi ćete napraviti posebnu stranicu za logiranje.

Napisat ćete stranicu login.php s IF strukturom i AKO korisnik nije 'ivica' ili ako zaporka nije '1234' poslat ćete poruku 'Netočni podaci!'. U suprotnom ćete pustiti korisnika prema traženoj stranici, na primjer admin.php. Ali tu zabranjenu stranicu mogu pronaći i tražilice, mogu slučajno ili namjerno pozvati i posjetitelji tipkanjem admin.php u address polje preglednika.

Isto tako, ukoliko imamo desetak stranica koje štitimo, nećemo se valjda morati desetak puta i logirati. Ne, nećemo, jer postoji sesija, o čemu čitajte u nastavku, a detaljno na adresi:

<http://www.php.net/manual/en/ref.session.php>

Sesija

Svaki put kad preko našeg klijenta pošaljemo zahtijev poslužitelju, poslužitelj kreira varijablu session id koja sadrži jedinstveni string koji obilježava vas kao jedinstvenog posjetitelja. Kad poslužitelj pošalje stranicu prema vama, on obriše sve u memoriji, osim tog vašeg jedinstvenog broja. Dakle, pada u vodu ono što sam do sad govorio da web poslužitelj gubi svaki trag o vama. Ne, ne gubi, postoji taj broj. Taj broj se čuva u memoriji poslužitelja vrlo kratko vrijeme. Međutim, sve što sam rekao ranije o ne prepoznavanju vas od strane poslužitelja je također istina, jer ma koliko brzo vi opet kliknuli na novi uzastopni zahtijev, poslužitelj će vam svaki put dodijeliti drugi jedinstveni string, jer on ne zna da ste baš vi taj isti posjetitelj od maloprije. On bi to znao samo kad bi poslao taj jedinstveni broj vašem klijentu zajedno sa stranicom i kad bi klijent automatski taj broj vratio poslužitelju s vašim novim zahtijevom. Tad bi znao. I upravo tako radi sesija. Taj broj šeta tamo-amo između klijenta i poslužitelja.

Ta datoteka u kojoj je jedinstveni broj sesije prenosi se između klijenta i poslužitelja na dva načina:

1. kao cookie
2. ili preko URL tj. pomoću metoda GET ili POST

Konfiguracija poslužitelja je najčešće takva da ukoliko je na klijent strani onemogućena funkcija cookie, sesija se automatski događa preko url. Preferira se varijanta cookie jer je sigurnija od one preko url.

U našem primjeru koristimo cookie varijantu prijenosa, zato će primjer raditi samo ako je na klijent strani omogućena funkcija cookie.

Kad god koristimo cookie varijantu prijenosa sesije, moramo prije ičega što se vraća posjetitelju pisati funkciju:

```
session_start()
```

Funkcija **session_start()** kreira sesiju ili svaki put iznova uspostavlja već postojeću sesiju na osnovi tekućeg stringa id sesije, a koji se prenosi preko upita tipa GET, POST ili cookie.

Kako izgleda string sesije (SID) možete vidjeti pomoću ove stranice:

```
<?php
session_start();
$session_id = session_id();
echo $session_id;
?>
```

U mapi 'login' koju možete skinuti na adresi www.e92.hr je kompletna aplikacija 'Blog' plus stranice za logiranje. Evo tih stranica i njihovog kôda, te objašnjenja tog koda.

Najprije kôd za one koji žele tipkati ga i tako vježbati.

loginForm.php

```

<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>PHP</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
</head>
<body>
<form method="post" action="login.php">
<input type="text" name="korisnik" value="<?php echo
$korisnik;?>">
<input type="password" name="zaporka" value="<?php echo
$zaporka;?>">
<input type="submit" value="Login">
</form>
</body>
</html>

```

login.php

```

<?php
require_once("tajne_konstante.php");
$korisnik = $_POST['korisnik'];
$zaporka = $_POST['zaporka'];
if(($korisnik != ADMINISTRATOR ) || ($zaporka != ZAPORKA )) {
    include("loginForma.php");
    exit;
}
if (($korisnik == ADMINISTRATOR ) && ($zaporka == ZAPORKA ))
{
    session_start();
    $_SESSION['administrator'] = ADMINISTRATOR;
    $_SESSION['zaporka'] = ZAPORKA;
    $SID = session_id();
    $admin_stranica = ADMIN_STRANICA;
    include($admin_stranica);
}
?>

```

sigurnosniKod.php

```

<?php
session_start();
if( (!isset($_SESSION['administrator'])) ||
(!isset($_SESSION['zaporka'])) ) {
    include_once("login.php");
    exit;
}

require_once("tajne_konstante.php");

```

```

if( ($_SESSION['administrator'] != ADMINISTRATOR) ||
($_SESSION['zaporka'] != ZAPORKA) ) {
    include_once("login.php");
    exit;
}else{?>
<a href="<?php echo ADMIN_STRANICA;?>">Admin</a>
<a href="odjava.php">Odjava</a>
<?php }?>

```

odjava.php

```

<?php
session_start();
function session_clear() {
// ako sesija postoji, skini registraciju sa svih varijabli i
uništi sesiju
    $postoji_registrirana_varijabla = "da";
    $session_array = explode(";",session_encode());

    for ($x = 0; $x < count($session_array); $x++) {
        $ime_varijable = substr($session_array[$x], 0,
strpos($session_array[$x],"|"));

// echo $ime_varijable;exit;

        if (session_is_registered($ime_varijable)) {
            session_unregister('$ime_varijable');
            $postoji_registrirana_varijabla = "ne";
        }
    }
if ($postoji_registrirana_varijabla != "da") {
    session_destroy();
}
}
session_clear();
if(!session_is_registered(session_name())) {
    echo"<h1>Odjavljeni ste.</h1>";
}else{
    echo"Niste odjavljeni.</h1>";
}
?>

```

tajne_konstante.php

```

<?php
define("ADMINISTRATOR", "admin");
define("ZAPORKA", "admin");
define("ADMIN_STRANICA", "admin.php");
?>

```

```
<?php require_once("sigurnosniKod.php");?>
```

Ovaj sigurnosni kôd sam dodao na vrh svake php stranice koju želim zaštititi, tj. stranice kojoj može pristupiti samo logirani korisnik. To su u mom slučaju sve stranice u aplikaciji 'Blog' osim index.php, dodaj.php, dodaj_pohrani.php.

To znači da tekstove u Blog može dodavati tko god želi, a administrirati Blog može samo ovlaštena osoba. U našem slučaju je to osoba čiji su podaci za logiranje:

korisničko ime: admin

zaporka: admin.

Objašnjenje

loginForma.php

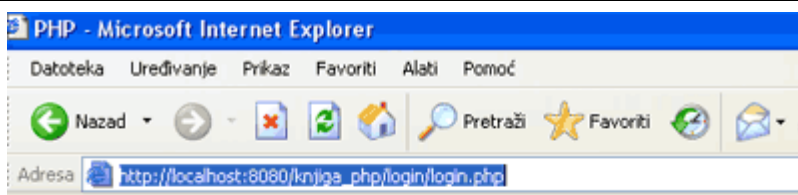
Ova stranica je mogla imati nastavak i .htm ili .html, jer sadrži samo formu, bez iti jednog retka php kôda:

```
<form method="post" action="login.php">
<input type="text" name="korisnik" value="<?php echo
$korisnik;?>">
<input type="password" name="zaporka" value="<?php echo
$zaporka;?>">
<input type="submit" value="Login">
</form>
```

Obratite pozornost na bitno u ovoj formi:

1. Kad kliknemo na gumb Submit (na njemu piše u našem slučaju 'Login'), bit će poslan upit prema poslužitelju. Upit će prenijeti adresu 'login.php', ali i vrijdnosti koje će korisnik tipkati u formu.
2. Taj upit će biti prenijet pomoću metode POST. To znači da se neće vidjeti vrijednosti koje ćemo tipkati u formu.
3. U formu ćemo tipkati 'admin' i 'admin' (korisničko ime i zaporka).
4. Klik na gumb Login i eto nas na adresi:

http://localhost:8080/knjiga_php/login/login.php



[Admin Odiava](#)

Administracija bloga

[Dodaj Početna](#) [Administracija](#)

Subjekt	Unos	Datum	Autor		
wtre	reter	2006-08-07	tertert	Obriši	Promijeni
sačlkdm	čldks	2006-08-07	sdčlclfd	Obriši	Promijeni
kasjda	jksdhnjks	2006-08-07	,msnds	Obriši	Promijeni
gdffg	fdgfdgf	2006-08-05	dgdgdf	Obriši	Promijeni
sdfsff	dfdsfdfsdfds	2006-08-05	fd sdfsfdsfdfs	Obriši	Promijeni

Napomena: Mi smo dobili na ekran stranicu admin.php, ali u adresi piše stranica login.php. Koja je to onda stranica? To je ona stranica koja piše u adresi, login.php. U nastavku je njen kôd pa ćete vidjeti zašto vidite stranicu admin.php. Vidite je radi one zadnje instrukcije na stranici login.php:

```
include($admin_stranica);
```

Funkcija 'include()' učitava kompletan kôd stranice koja je navedena u zagradama. U našem slučaju je u zagradi varijabla \$admin_stranica čija je vrijednost admin.php.

login.php

```
<?php
```

Funkcija 'require_once()', baš kao i funkcija 'include()', učitava kompletan kôd one stranice koja je navedena u zagradama. U našem slučaju je to stranica na kojoj su definirane konstante za logiranje.

```
require_once("tajne_konstante.php");
```

Kreiramo varijable \$korisnik i \$zaporka i dajemo im vrijednosti globalnih varijabli \$_POST['korisnik'] i \$_POST['zaporka']. Globalne varijable \$_POST[] su 'uhvatile' 'admin', string kojeg smo tipkali u <INPUT name="korisnik" value="admin"> i <INPUT name="zaporka" value="admin">.

```
$korisnik = $_POST['korisnik'];
$zaporka = $_POST['zaporka'];
```

Ako vrijednosti varijabli \$korisnik i \$zaporka NISU (!=) jednake 'admin', 'admin', jer ADMINISTRATOR je konstanta kojoj smo dali vrijednost 'admin', a ZAPORKA konstanta vrijednosti 'admin', onda će se učitati programski kod stranice loginForma.php, i zaustavit će se daljnje procesuiranje (exit):

```
if(( $korisnik != ADMINISTRATOR ) || ( $zaporka != ZAPORKA ) ) {
    include("loginForma.php");
    exit;
}
```

SAMO ako su obje varijable jednake vrijednosti 'admin', kreirat će se sesija. Znak '&&' znači "logičko i". Ako pogledate u tablicu istine za LOGIČKO I, vidjet ćete da će logička konstrukcija A && B, kao što je naša, imati vrijednost '1' ili 'true' SAMO ako su A=1 i B=1. U našem slučaju će A biti 1 samo ako je \$korisnik == ADMINISTRATOR (admin), a B će biti 1 samo ako je \$zaporka == ZAPORKA (admin).

NAPOMENA: Razlika između jednog znaka jednakosti = i dva znaka jednakosti je sljedeća. Jedan znak = znači da lijevoj strani DAJEMO vrijednost desne strane. Dva znaka == je logička funkcija kojom TVRDIMO da je lijeva strana jednaka desnoj. Takva tvrdnja uvijek izbaci 1 ili 0. Ako je istinita izbaci 1, a ako je laž izbaci 0.

Ponavljam: iza IF, uvijek u okruglim zagradama, slijedi TVRDNJA ili više TVRDNJI povezanih logičkim operatorima kao što su ==, !=, &&, || (ili 'or'). Cijela okrugla zagrada, ma koliko tvrdnji u njoj bilo, uvijek je u konačnici jednaka 1 ili 0.

Preporučujem: napraviti što je moguće više vježbi s ovim logičkim funkcijama i konstrukcijom IF.

```
if (( $korisnik == ADMINISTRATOR ) && ( $zaporka == ZAPORKA ) )
{
    session_start();
    $_SESSION['administrator'] = ADMINISTRATOR;
    $_SESSION['zaporka'] = ZAPORKA;
    $SID = session_id();
    $admin_stranica = ADMIN_STRANICA;
    include($admin_stranica);
}
?>
```

Sesiju preko cookieja kreiramo s funkcijom session_start(). Varijable koje će se pohraniti u automatski kreiranu tablicu session na serveru, formiramo jednostavnim ispisom:

```
$_SESSION['administrator'] = ADMINISTRATOR;
$_SESSION['zaporka'] = ZAPORKA;
```

Funkcija session_id() izbaci string sesije kojeg pohranjujemo u varijablu \$SID:

```
$SID = session_id();
```

Kreiramo varijablu koja će pohraniti vrijednost konstante ADMIN_STRANICA. U našem slučaju je ta vrijednost admin.php.

```
$admin_stranica = ADMIN_STRANICA;
```

Pomoću funkcije INCLUDE() učitavamo programski kôd stranice admin.php, koju ćemo onda i vidjeti na ekranu, iako će u address polju pisati ...login.php. Jer točno

je, na ekranu je stranica login.php, ali toj stranici jedini HTML dolazi upravo od učitane stranice admin.php. Stranica admin.php se ispisala na stranici login.php.

```
include($admin_stranica);
```

Zaključak: vrlo je lako upravljati sa sesijom. Samo napišemo funkciju

```
session_start();
```

i kreiramo varijable sesije po volji:

```
$_SESSION['administrator'] = ADMINISTRATOR;
$_SESSION['zaporka'] = ZAPORKA;
```

sigurnosniKod.php

Ova stranica je 'čarobni štapić'. Pozovemo je na početku svake stranice koju štitimo i to je sve:

```
<?php require_once("sigurnosniKod.php");?>
```

Što radi taj 'čarobni štapić'?

Na početku moramo staviti funkciju session_start(). Na taj način smo kreirali novu sesiju ili osvježili postojeću ako postoji.

```
<?php
session_start();
```

Na ovom mjestu kôda aplikacije SESIJA ili postoji ili ne postoji. Funkcija session_start() je obavila svoj posao, a taj je osvježavanje postojeće SESIJE ako postoji, ili priprema baze za novu sesiju koja će na ovom mjestu biti prazna, jer još nismo definirali varijable. Niti ćemo. Nas sad zanima samo postoji li sesija ili ne. To provjeravamo preko logičkih funkcija i strukture IF, a uz pomoć naših varijabli koje moraju postojati ako se korisnik uspješno logirao. IF očekuje 1 ili 0 u zagradama. Koristimo funkciju ISSET() koja izbacuje 1 ili 0. Ako POSTOJI ono što je u zagradi funkcije isset, funkcija izbacuje 1. Ako ne postoji, izbacuje 0. Mi ovdje koristimo logički operator ! (ISSET) koji znači NE. Imamo logičku konstrukciju A ILI B koja se piše A || B (okomite crte pišete s ALT+0124). Logički izraz 'A || B' će biti 1 UVIJEK kad je A ili B jednako 1, ili oboje jednako 1. Bit će 0 samo ako su i A i B jednaki 0.

Zadatak za vas: Napišite tablicu istine za A || B, pa onda tablicu istine za logički izraz (!A || !B)

Moja pomoć vama: logički izraz (!A || !B), koji u našem primjeru izgleda ovako:

```
( (!isset($_SESSION['administrator'])) ||
  (!isset($_SESSION['zaporka'])) )
```

se sastoji od DVIJE TVRDNJE. Prva TVRDNJA glasi,

```
!isset($_SESSION['administrator'])
```

riječima 'NE POSTOJI u tablici sesija varijabla administrator'. Ta tvrdnja će izbaciti 1 ako je ISTINITA.

Druga TVRDNJA glasi,

```
!isset($_SESSION['zaporka'])
```

riječima 'NE POSTOJI u tablici sesija varijabla zaporka'. Ta tvrdnja će izbaciti 1 ako je ISTINITA. Logička konstrukcija $A \parallel B$ izbacuje 1 ako je A ili B ili A i B jednako 1.

```
if( (!isset($_SESSION['administrator'])) ||
    (!isset($_SESSION['zaporka'])) )
```

Tako smo došli do naše jednostavne želje da se na ekranu posjetitelja pojavi FORMA za logiranje ako ne postoji jedna ili obje potrebne varijable u tablici sesije. Kôd ne poziva direktno stranicu loginForma, nego zove stranicu login.php, a ova će pozvati stranicu loginForma.php.

```
{
    include_once("login.php");
    exit;
}
```

Ako obje varijable postoje, idu na provjeru. Prije provjere se učitava stranica koja na sebi ima definirane konstante, potrebne za provjeru valjanosti varijabli, tj. provjeru da li je korisnik tipkao vrijednosti 'admin', 'admin'.

```
require_once("tajne_konstante.php");
```

Evo opet konstrukcije IF kojom vršimo tu provjeru. Opet je logički izraz $A \parallel B$. Ako jedna od varijabli nije jednaka tajnoj vrijednosti 'admin', ponovo se poziva forma indirektno preko stranice login.php.

```
if( ($_SESSION['administrator'] != ADMINISTRATOR) ||
    ($_SESSION['zaporka'] != ZAPORKA) ) {
    include_once("login.php");
    exit;
}
```

U suprotnom, izbaciti se ovaj HTML:

```
}else{?>
<a href="<?php echo ADMIN_STRANICA;?>">Admin</a>
<a href="odjava.php">Odjava</a>
<?php }?>
```

Sjetimo se od čega smo počeli. Na vrh zaštićenih učitavamo upravo opisani kôd:

```
<?php require_once("sigurnosniKod.php");?>
```

Ukoliko postoji ispravna sesija, taj kôd će izbaciti samo dva linka:

```
<a href="<?php echo ADMIN_STRANICA;?>">Admin</a>
<a href="odjava.php">Odjava</a>
```

Ispod ta dva linka će se pojaviti HTML stranice na čijem vrhu smo napisali:

```
<?php require_once("sigurnosniKod.php");?>
```

odjava.php

Moramo početi s funkcijom session_start() koja će osvježiti postojeću sesiju. Tko poziva tu funkciju? Nisam napisao nigdje kôd koji poziva tu funkciju. Svaka funkcija će se procesuirati SAMO ako je netko pozove. Odgovor: pozvana je automatski od baze sesija i to je automatizam kreiranja nove ili osvježavanja postojeće sesije.

```
<?php
session_start();
```

Na ovom mjestu kôda se osvježila baza sesije na poslužitelju i u njoj sve registrirane varijable. Funkciju `session_clear()`, koju koristimo u postupku uništenja sesije, također poziva automatski baza sesije, i ona se izvršava.

```
function session_clear()
```

Uloga funkcije `session_clear()` je skinuti registracije s registriranih varijabli u bazi. Jer samo ako su u bazi sve varijable NE-registrirane, tablica se može uništiti.

Kreiramo pomoćnu varijablu `$postoji_registrirana_varijabla`, koja služi samo kao prekidač ili kao zastavica koja će nam javiti kad više ne bude niti jedne registrirane varijable.

```
{
    $postoji_registrirana_varijabla = "da";
```

Pomoću funkcije `EXPLODE()` i funkcije `SESSION_ENCODE()`, varijable se izdvoje iz baze sesije i memoriraju se u array strukturu (varijablu) kojoj smo dali ime `$session_array`:

```
$session_array = explode(";",session_encode());
```

Pomoću PETLJE FOR smo iz arraya pozivali redom jednu po jednu registriranu varijablu...

```
for ($x = 0; $x < count($session_array); $x++) {
    $ime_varijable = substr($session_array[$x], 0,
    strpos($session_array[$x], "|"));
```

...čije ime možete i vidjeti ako skinete znak komentara //

```
// echo $ime_varijable;exit;
```

Pomoću strukture IF i funkcije `SESSION_IS_REGISTERED()` provjeravamo da li je varijabla registrirana.

```
if (session_is_registered($ime_varijable)) {
```

Ako je registrirana, pomoću funkcije `SESSION_UNREGISTER()` uništavamo tu registraciju

```
    session_unregister('$ime_varijable');
```

i stavljamo zastavicu na vrijednost 'ne':

```
        $postoji_registrirana_varijabla = "ne";
    }
}
```

Još jedna provjera prije konačnog uništenja sesije:

```
if ($postoji_registrirana_varijabla != "da") {
```

Konačno uništenje sesije pomoću funkcije `SESSION_DESTROY()`

```
    session_destroy();
}
}
```

Još jednom funkcija `session_clear()`

```

        session_clear();
pa funkcija SESSION_IS_REGISTERED()
        if(!session_is_registered(session_name())) {
do konačne poruke 'Odjavljeni ste'
                echo"<h1>Odjavljeni ste.</h1>";
ili
        }else{
niste
                echo"Niste odjavljeni.</h1>";

        }
?>

```

tajne_konstante.php

Ovdje su definirane konstante:

```

<?php
define("ADMINISTRATOR", "admin");
define("ZAPORKA", "admin");
define("ADMIN_STRANICA", "admin.php");
?>

```

Kad želimo promijeniti korisničko ime ili zaporku, to činimo ovdje.

Zaključak

Aplikaciju LOGIN koristimo na ovaj način.

1. Na stranici tajne_konstante.php upišemo korisničko ime, zaporku i ciljnu stranicu.
2. Na vrh kôda svake zaštićene stranice kopiramo ovaj kôd:

```
<?php require_once("sigurnosniKod.php");?>
```

I to je sve!

Zadaci

1. Dodajte aplikaciju LOGIN svojim PHP aplikacijama.
2. Formatirajte aplikaciju LOGIN pomoću CSS-a i promijenite strukturu stranica pomoću XHTML.
3. Želite na svakoj svojoj stranici link NAZAD. Taj link mora biti dinamički jer svakoj stranici najčešće možete pristupiti linkovima odasvud iz svoje aplikacije. Kako će link NAZAD znati odakle ste vi došli? Sesija je rješenje. Jedan primjer takvog rješenja možete vidjeti u aplikaciji CMS kasnije u knjizi.
4. Ovaj zadatak je samo za one koji znaju klijentski program JavaScript. Formi za logiranje dodajte JavaScript kôd koji će element <INPUT

name="korisnik"> staviti u fokus.

Rješenje: na stranici loginForma.php dodajte boldirani kôd:

```
<body onload="window.document.loginForma.korisnik.focus()">  
<form name="loginForma" method="post" action="login.php">
```

Napomena: ovaj mali dodatak kôda koji se izvršava na klijentskoj strani, učinio je puno na polju uporabljivosti (usability) grafičkog sučelja vaše aplikacije.

5. Napišite tablicu istine za sljedeće logičke izraze:
 - a. $A \parallel B$
 - b. $!A \parallel !B$
 - c. $!A \parallel B$
 - d. $A \parallel !B$
 - e. $(A \parallel B) \&\& (A \&\& B)$
 - f. $A = 1$ i $B = 0$. Što će izbaciti logički izraz: $(A \parallel B) \&\& (!A \&\& B)$
 - g. $A = 1$ i $B = 1$. Što će izbaciti logički izraz: $(!A \parallel !B) \&\& (!A \&\& B)$
 - h. $A = 1$ i $B = 1$. Što će izbaciti logički izraz: $(!A \parallel !B) \parallel (A \&\& !B)$
6. Učenici koji učite online, zadajte jedni drugima logičke zadatke na forumu ili chatu.

Test

7. Što je sesija?
8. Što je to SID?
9. Gdje se kreira SID?
10. Kad se kreira SID?
11. Kako se sesija prenosi između klijenta i poslužitelja?
12. Opiši postupak uništavanja sesije.
13. Koju funkciju moramo pozvati na početku svakog kôda ukoliko želimo kreirati novu sesiju ili osvježiti postojeću?
14. Na ekranu je forma za logiranje. Upisali ste krive podatke i kliknuli na gumb Login. Što će se dogoditi na ekranu?
15. Na ekranu je forma za logiranje. Upisali ste krive podatke i kliknuli na gumb Login. Krivi podaci će ostati upisani u poljima. Koji redak kôda je za to zaslužan?
16. Opišite cijeli postupak logiranja.



PHP Objektno orijentirano programiranje

Metodika-----

Objektno orijentirano programiranje je današnje profesionalno programiranje. Proizvodnja aplikacija za tržište danas je moguća samo u OOP. Zašto? Evo mojih iskustava.

- Objektno orijentirano programiranje podrazumijeva, prije svega, jednu izvrsnu organiziranost aplikacije, a onda i procesa kreiranja iste.
- Objektno orijentirano programiranje znači da je riječ o vrhunskom profesionalcu programeru, pa je za očekivati i vrhunski proizvod.
- Tako dobro organizirana aplikacija je lako i jeftino nadograđivana, lako i jeftino održavana, unaprijeđivana. I to ne samo od strane autora, nego od strane bilo kojeg programera.
- I za kraj, ovdje ću još reći ovo. Ta fenomenalna podijela na apstraktno i konkretno je zaslužna za sve što sam nabrojio gore. Klasa je apstrakcija, objekt je konkretnost. O tome više u nastavku.

Ovu sam knjigu zamislio tako što ću u nju pisati svoje misli, ideje, razmišljanja, iskustva. Davno je to bilo kad je autor želio da mu knjiga bude jedan kompletan rad u kojem će čitatelji naći sve na tu temu. Ne. Toga više nema. Ako vam je dovoljan jedan klik na www.php.net za pronaći OPERATORE ili VARIJABLE ili OBJEKTE i KLASU, itd., zašto bih ja to isto pisao u knjigu i učinio je ne pristupačnom po cijeni, a sigurno ne konkurentnoj samo jednoj adresi na Internetu: www.php.net od tisuće i tisuće drugih ako se obratite Googleu.

Na Internetu ne piše kako sam ja radio prošlu školsku godinu.

Zato je sad lakše pisati knjige. Taj dosadni štreberski dio uvijek možemo pozvati linkovima. Kreativni dio ostaje nama i našoj knjizi.

Isti koncept vrijedi i za pripremu i izvođenje nastave danas. Na satu treba vježbati, a sve podatke vezane za sintaksu, operatore, tipove varijabli, kako ovo, kako ono, treba riješavati klikom na link.

Zato je u ovoj nastavnoj jedinici naglasak na postizanju razumijevanja što je to klasa a što objekt i sve snage treba usmjeriti k tom cilju.

Vježbajući treba se priviknuti na famozno pozivanje članova objekta pomoću pseudo objekta \$this i strelice ->.

U klasi smo ispisali sve što će budući objekti bazirani na njoj raditi. Kad se naša klasa učitava u RAM, ništa se ne događa. Program je živ, ali kao da i nije. Klase su "mrtvo slovo na papiru" koje zauzimaju memoriju i čine dio sveukupne aplikacije.

```
include "MojaPrvaKlasa.php";
```

Tek kad se pojavi ovaj kôd:

```
$objekt = new MojaPrvaKlasa();
```

počelo se i nešto KONKRETNO događati. Sljedi to događanje:

```
echo $objekt->javnaVarijabla;
echo $objekt->zasticenaVarijabla;
echo $objekt->privatnaVarijabla;
```

posljedica čega je konkretna pojava konkretnog sadržaja na našoj web stranici. Ono što nam se vrati na ekran, nakon našeg zahtijeva, to je konkretnost, to je rezultat rada našeg PHP programa, naše PHP aplikacije.

Treba dobro pojmiti razliku između INSTANCIRANJA i INICIJALIZIRANJA.

Sinonimi. Na www.php.net i drugdje naći ćete različite izraze za jedno te isto. Evo nama najvažnijih:

- članovi objekta = varijable objekta = svojstva (properties) objekta
- metode objekta = funkcije objekta

Konstruktor. To je metoda koja se izvrši u instrukciji instanciranja objekta, dakle u ovoj instrukciji:

```
$ivica = new Student("Ivica", "Marin", "ivica.gif");
```

I sad zamislite ovu situaciju:

- Napravili ste klasu 'test' koja sadrži konstruktor.
- U konstruktoru ste napisali jedan poveći program koji će imati i interakcije s posjetiteljem. Na primjer, izbacivat će forme s pitanjima, posjetitelj će odgovoriti, ovisno o odgovoru dobit će drugo pitanje i tako na primjer sat vremena. Neki veliki test ili anketa.
- Klasa je napisana. Kako će izgledati vaša PHP aplikacija? Ovako:


```
<?php
include "test.php";
$test = new Test();
?>
```
- Zar to nije fenomenalno!? Aplikacija koja traje sat vremena, koja testira našeg učenika ili provodi anketu, ima samo DVIJE INSTRUKCIJE! U jednoj instrukciji se u RAM učitala APSTRAKCIJA, u drugoj instrukciji se POKRENULA KONKRETIZACIJA te apstrakcije. Prvi redak je tek IDEJA. Drugi redak je REALIZACIJA te ideje.

Naslijeđivanje (extends) je važan dio OOP. Važan je baš sa stajališta DOBRE ORGANIZIRANOSTI projekta. Možemo napraviti klasu WebStranica koja će imati ono što MORA imati svaka web stranica kad je radimo u XHTML-u. A onda

možemo napraviti klasu Naslovnica koja će naslijediti sve članove i metode koje ima WebStranica, plus što ćemo joj dodati one elemente koje ima samo ta stranica. Nema dobre organizacije, bez dobre hijerarhije, ma koliko bila plitka ili duboka, i mora je otkriti i kreirati programer.

Naknadne intervencije u program su jednostavne. Ako želimo svim stranicama nešto izmjeniti, napraviti ćemo to na onoj najstarijoj po hijerarhiji, klasi WebStranica.

Udžbenik-----

Što ćemo naučiti?

1. Klasa
2. Objekt
3. Vidljivost
4. Nasljedstvo
5. Operator ::

Klasa

Osnovna ideja na kojoj počiva OOP je ispisati programski kôd u jednu cjelinu koja ima svoje ime, pa kad nam god zatreba ta cjelina, pozvati je preko njenog imena i pustiti u rad. Ali, zar to već nemamo s funkcijama tipa include()? Imamo, ako bismo ostali samo na tome. Međutim, od te osnovne ideje dogodila se ogromna evolucija OOP, tako da je ne usporediva s funkcijom include(). Dogodilo se to da je ta programska cjelina postala predložak kojeg zovemo klasa.

Čiji predložak? Predložak za objekte. Objekti su tako postali 'živi' igrači u runtimeu programa, a klasa je samo mrtvo slovo na papiru. Nacrt kuće i stvarna kuća. Ovo prvo je klasa, a ovo drugo je objekt. Objekti u programu su oni koji rade. Klasa je APSTRAKCIJA, a objekt je KONKRETNOST.

Josip Kumerički, savjetnik za razvoj u Samoborki d.d. ovako kaže:

1. APSTRAKCIJA - pojam, stvaran u procesu upoznavanja konkretnosti, i predstavlja rezultat misaonog djelovanja –umnog rada. To je odraz konkretnosti u ljudskom saznanju.
2. KONKRETNOST - bilo koji materijalni objekt prirodni ili umjetni. Konkretno znači materijalni, a ne npr. «dani» ili «strogo opredjeljen». Konkretna djelatnost, to je rad, djelovanje konkretne osobe.

Netko prijavi svoj izum patentnom zavodu. To je APSTRAKCIJA. Netko primjeni taj izum, to je KONKRETNOST."

Hvala Josipu na pomoći. Znači, klasa je APSTRAKCIJA, a objekt je KONKRETNOST.

A onda su ti objekti postali roditelji, pa su došla objekti djeca, pa nasljeđe... Programerska priča je postala preslika svijeta ljudi. Zahvaljujući tome, objektno orijentirano programiranje postalo je zakon. Neusporedivo je razumljivije, lakše za naučiti, lakše za održavati, lakše za razvijati. Programer je *učinkovitiji i djelotvorniji*.

Joža Kumerički o učinkovitosti i djelotvornosti kaže sljedeće: "Mnogi slavni umjetnici su za života bili više gladni nego siti. To je loša učinkovitost. Danas im se djela prodaju za milijune dolara, to je djelotvornost. Industrijsko društvo se pokazalo puno učinkovitije od agrarnog po rastu standarda stanovništva. Koliko je pri tome oštetilo okoliš, to se pripisuje DJELOTVORNOSTI".

Veće aplikacije su danas nezamislive bez 100% OOP. Jer, OOP aplikacije vrlo lako koristite i godinama nakon njihova nastanka. Tako je programer puno djelotvorniji ako radi OOP. Međutim, u početku, programer može biti čak i manje učinkovit ako radi OOP, tj. potrošiti više vremena za istu aplikaciju nego je trošio do tad. To je zato što u početku nema zaliha gotovih klasa koje može upotrijebiti, a kojih će sve više imati. PONOVDNA UPOTREBLJIVOST je glavna odlika OOP, a to je daleki san naše materijalne stvarnosti.

Moja prva klasa

1. Tipkajte u Notepad programski kôd 'MojaPrvaKlasa.php':

```
<?php
class MojaPrvaKlasa
{
    public $javnaVarijabla = 'Javna varijabla';
    protected $zasticenaVarijabla = 'Zaštićena varijabla';
    private $privatnaVarijabla = 'Privatna varijabla';

    public function TiskajVarijable()
    {
        echo $this->javnaVarijabla;
        echo $this->zasticenaVarijabla;
        echo $this->privatnaVarijabla;
    }
}
?>
```

2. Pohranite pod imenom MojaPrvaKlasa.php, pa je otvorite u web pregledniku.
Prazna web stranica! Ispravno. Sjetimo se, klasa 'ne igra na terenu'. Klasa je samo predložak onima koji će 'igrati na terenu', a to su objekti.

Objašnjenje klase

Klasa počinje s ključnom riječi class. Sljedi razmaknica pa ime klase:

```
class MojaPrvaKlasa
```

U vitičastim zagradama pišemo programski kôd klase.

```
{
```

Svojstva (properties) klase (ili članovi klase, ili varijable klase)

Klasa ima članove. Naša klasa ima ova tri člana: \$javnaVarijabla, \$zasticenaVarijabla, \$privatnaVarijabla. Član ima ispred svog imena znak dolara \$. Ispred člana ide jedna od tri ključne riječi: public, protected, private. Te tri riječi označavaju vidljivost članova (dohvatljivost članova ili životni vijek članova). Član kojeg deklariramo 'public', dohvatljiv (ili vidljiv) je svugdje i od svakuda na stranici. 'Protected' je dohvatljiv samo roditeljima i nasljednicima, a 'private' je vidljiv samo svojoj klasi.

```
public $javnaVarijabla = 'Javna varijabla';
protected $zasticenaVarijabla = 'Zaštićena varijabla';
private $privatnaVarijabla = 'Privatna varijabla';
```

Metode

Klasa ima metode. Metode su funkcije. I funkcije se deklariraju s public, protected i private i značaj tih deklaracija je isti kao za članove. Ako nismo stavili deklaraciju, PHP će smatrati tu funkciju public. Funkcije ćete uvijek prepoznati po zagradama koje slijede odmah iza imena funkcije. Zagrade mogu biti prazne ili pune parametara (može se reći i argumenata) preko kojih se prenose vrijednosti na varijable unutar funkcije. Evo funkcije u našoj klasi:

```
public function TiskajVarijabla()
{
```

\$this

Instrukcija koja počinje s ključnom riječi 'echo' nam je poznata. Ovdje je novost sintaksa \$this->javnaVarijabla. '\$this' znači 'ova klasa', strelica -> znači 'poziva' i u drugim tehnologijama na primjer .NET i Java je to točka (.). Uvijek kad pozivamo člana ili metodu, pozivamo ih pomoću strelice ->.

Dakle, \$this->javnaVarijabla znači 'ova klasa poziva svog člana javnaVarijabla'. Samo pomoću ovog pseudonima \$this se mogu 'izvući na vidjelo' članovi. Umjesto riječi pseudonim, puno preciznije je reći: \$this znači 'ja'. Ja (klasa) pozivam svog člana Obratite pozornost: članove smo ovdje pisali bez znaka dolara (\$) ispred njihova imena.

```
echo $this->javnaVarijabla;
echo $this->zasticenaVarijabla;
echo $this->privatnaVarijabla;
}
}
?>
```

I to je sve o našoj prvoj klasi, ali i o svim budućim koje ćemo kreirati.

Napravimo svoj prvi objekt.

Objekt

Moj prvi objekt

1. Tipkajte u Notepad programski kôd 'MojPrviObjekt.php':

```
<?php
include "MojaPrvaKlasa.php";

$objekt = new MojaPrvaKlasa();
echo $objekt->javnaVarijabla;
echo $objekt->zasticenaVarijabla;
echo $objekt->privatnaVarijabla;

?>
```

2. Pohranite pod imenom MojPrviObjekt.php, otvorite u web pregledniku i dobit ćete ovo na web stranici:

```
Javna varijabla
```

Objašnjenje objekta

Pomoću funkcije include učitali smo programski kôd naše prve klase. Klasa je tako dospijela u memoriju poslužitelja i spremna je poslužiti kao predložak objektima.

```
<?php
include "MojaPrvaKlasa.php";
```

Instanciranje objekta

I evo našeg prvog objekta. Kreiramo našu prvu, novu instancu objekta. Ime objekta je kao ime varijable, sa znakom dolara ispred. Slijedi znak jednako = s razmacima oko sebe ili bez razmaka, kako vam je preglednije. Na desnoj strani znaka = je ključna riječ 'new', slijedi razmaknica i ime klase. Na kraju imena klase su uvijek i zagrade kao kod funkcija, u koje možemo pisati parametre. S tom instrukcijom smo *instancirali* objekt:

```
$objekt = new MojaPrvaKlasa();
```

Inicijaliziranje objekta

Inicijalizirati objekt znači dodijeliti mu prvu vrijednost. U našem primjeru se to događa u ovoj instrukciji:

```
echo $objekt->javnaVarijabla;
```

Ta instrukcija doslovce glasi ovako u slobodnom prijevodu: tiskaj na web stranicu \$objekt koji poziva člana javnaVarijabla.

Objekt poziva svojstvo (properties)

Sjedeće dvije instrukcije, istom sintaksom žele tiskati na web stranicu preostala dva člana, koje naš objekt može pozvati:

```
echo $objekt->zasticenaVarijabla;
echo $objekt->privatnaVarijabla;
```

```
?>
```

Međutim, na ekranu će se pojaviti samo vrijednost člana kojeg smo u klasi deklarirali 'public'.

Objekt poziva metodu

objekt2.php

```
<?php
include "MojaPrvaKlasa.php";

$objjekt = new MojaPrvaKlasa();
echo $objjekt->TiskajVarijable();

?>
```

Stranica će prikazati ovo:

```
Javna varijablaZaštićena varijablaPrivatna varijabla
```

Objašnjenje

Funkciju smo deklarirali 'public' i ona je prikazala i zaštićenu varijablu i privatnu varijablu, jer to nalaže njezin kôd, bez obzira što su zaštićene i privatne. Znači, programer je taj koji mora voditi računa o ključnim riječima 'public', 'protected' i 'private'.

Konstruktor

Konstruktor je funkcija koja će se izvršiti automatski čim INSTANCIRAMO objekt. Sljedi primjer.

student.php

```
<?php
class Student
{
    public $ime;
    public $prezime;
    public $slika;

    function __construct($imeStudenta, $prezimeStudenta,
        $slikaStudenta)
    {
        echo $this->ime = $imeStudenta . " ";
        echo $this->prezime = $prezimeStudenta;
        echo $this->slika = "<br /><img src='" . $slikaStudenta . "'
        /><br />";
    }
}

?>
```

studenti.php

```
<?php
include "student.php";
$ivica = new Student("Ivica", "Marin", "ivica.gif");
$vinko = new Student("Vinko", "Marin", "vinko.gif");
$marica = new Student("Marica", "Marin", "marica.gif");
?>
```

Objašnjenje

student.php

Najprije smo kôdirali APSTRAKTNE podatke – klasu student.

```
<?php
class Student
{
```

Deklariramo tri javna svojstva.

```
    public $ime;
    public $prezime;
    public $slika;
```

Konstruktor počinje kao svaka funkcija s ključnom riječi function, razmaknica, dvije donje crtice '_', ime construct i okrugle zagrade. Preko parametara u zagradama, mi, kao netko izvana, dodijelit ćemo vrijednosti varijablama unutar konstruktora.

```
    function __construct($imeStudenta, $prezimeStudenta,
        $slikaStudenta)
```

U vitičastim zagradama slijede instrukcije funkcije.

```
{
```

Instrukcija koja slijedi radi sljedeće. Piši na web stranicu svojstvo ime. Svojstvu ime dodijeli vrijednost varijable \$imeStudenta i razmaknicu \$imeStudenta . " ". Točka između stringova znači dodavanje stringova. Oko stringa se uvijek pišu navodnici, a oko varijabli se ne pišu.

```
    echo $this->ime = $imeStudenta . " ";
    echo $this->prezime = $prezimeStudenta;
    echo $this->slika = "<br /><img src='" . $slikaStudenta . "'
    /><br />";
    }
}
```

studenti.php

Ovdje je apstraktno (klasa) prešlo u konkretno (objekt).

```
<?php
```

Najprije smo učitali APSTRAKTNO (klasu) student.php:

```
include "student.php";
```

INSTANCIRAMO tri objekta:

```
$ivica = new Student("Ivica", "Marin", "ivica.gif");
$vinko = new Student("Vinko", "Marin", "vinko.gif");
$marica = new Student("Marica", "Marin", "marica.gif");
?>
```

Kad otvorimo studenti.php, vidjet ćemo

Ivica Marin

(slika)

Vinko Marin

(slika)

Marica Marin

(slika)

Objašnjenje

Dovoljno je bilo **instancirati** objekt pa da vidimo podatke na stranici. Nismo morali posebno pozivati funkciju `__construct`. Ta funkcija se uvijek automatski izvodi tijekom instanciranja objekata. Zato je konstruktor pogodan za svaku automatsku **inicijalizaciju** objekta prije početka upotrebe.

Klasa nasljednik

Klasa nasljednik svih svojstava i metoda neke klase, kreira se pomoću ključne riječi **extends**.

studentnasljednik.php

```
<?php
class NasljednikKlaseStudent extends Student
{
    public $datumRodjenja;
    public $brojIndeksa;
    public $upisaniFakultet;

    public function PodaciOStudentu($datumRodjenja, $brojIndeksa,
    $upisaniFakultet)
    {
        echo $this->datumRodjenja = "Godina rođenja: " .
        $datumRodjenja . "<br />";
        echo $this->brojIndeksa = "Broj indeksa: " . $brojIndeksa .
        "<br />";
        echo $this->upisaniFakultet = "Fakultet: " .
        $upisaniFakultet . "<br />";
    }
}
?>
```

studenti_nasljednici.php

```
<?php
```



```
include "student.php";
include "studentnasljednik.php";

$Ivica = new NasljednikKlaseStudent("Ivica", "Marin",
    "ivica.gif");
$Ivica->PodaciOStudentu("1950", "45455234", "FESB");
echo "<hr />";
$Vinko = new NasljednikKlaseStudent("Vinko", "Marin",
    "vinko.gif");
$Vinko->PodaciOStudentu("1980", "8978934214", "Ekonomski");
echo "<hr />";
$Marica = new NasljednikKlaseStudent("Marica", "Marin",
    "marica.gif");
$Marica->PodaciOStudentu("1988", "3123345438974", "Pravni");
echo "<hr />";
?>
```

Objašnjenje

studentnasljednik.php

Ovo je klasa koja ima svoja svojstva i metodu i koja uz to nasljeđuje sve metode i sva svojstva klase Student. Klasa nasljednik se kreira pomoću ključne riječi **extends**.

```
<?php
class NasljednikKlaseStudent extends Student
{
```

Svojstva klase.

```
    public $datumRodjenja;
    public $brojIndeksa;
    public $upisaniFakultet;
```

Metoda klase.

```
    public function PodaciOStudentu($datumRodjenja, $brojIndeksa,
        $upisaniFakultet)
    {
        echo $this->datumRodjenja = "Godina rođenja: " .
            $datumRodjenja . "<br />";
        echo $this->brojIndeksa = "Broj indeksa: " . $brojIndeksa .
            "<br />";
        echo $this->upisaniFakultet = "Fakultet: " .
            $upisaniFakultet . "<br />";
    }
}
?>
```

studenti_naljednici.php

Ovdje su objekti. Najprije učitamo klasu roditelja i klasu dijete.

```
<?php
include "student.php";
include "studentnasljednik.php";
```

Prijenos vrijednosti preko parametara klase u varijable konstruktora

Instanciranje (kreiranje) objekta \$ivica i dodjeljivanje vrijednosti varijablama konstruktora, preko parametara klase.

```
$ivica = new NasljednikKlaseStudent("Ivica", "Marin",  
"ivica.gif");
```

Prijenos vrijednosti preko parametara funkcije u varijable funkcije

Pozivanje funkcije i preko parametara dodjeljivanje vrijednosti varijablama unutar te funkcije:

```
$ivica->PodaciOStudentu("1950", "45455234", "FESB");  
echo "<hr />";
```

Instanciranje (kreiranje) objekta \$vinko:

```
$vinko = new NasljednikKlaseStudent("Vinko", "Marin",  
"vinko.gif");
```

Pozivanje funkcije i preko parametara dodjeljivanje vrijednosti varijablama unutar te funkcije:

```
$vinko->PodaciOStudentu("1980", "8978934214", "Ekonomski");  
echo "<hr />";
```

Instanciranje (kreiranje) objekta \$marica:

```
$marica = new NasljednikKlaseStudent("Marica", "Marin",  
"marica.gif");
```

Pozivanje funkcije i preko parametara dodjeljivanje vrijednosti varijablama unutar te funkcije:

```
$marica->PodaciOStudentu("1988", "3123345438974", "Pravni");  
echo "<hr />";  
?>
```

Zaključak

Instanciranje i inicijaliziranje objekata u ovom primjeru se dogodilo u jednoj jedinoj instrukciji, npr.:

```
$ivica = new NasljednikKlaseStudent("Ivica", "Marin",  
"ivica.gif");
```

Ovo će biti prikazano na web stranici kad otvorimo stranicu studenti_nasljednici.php

```
Ivica Marin  
(slika)  
Godina rođenja: 1988  
Broj indeksa: 45455234  
Fakultet: FESB
```

```
Vinko Marin  
(slika)
```

Godina rođenja: 1980
Broj indeksa: 8978934214
Fakultet: Ekonomski

Marica Marin
(slika)
Godina rođenja: 1988
Broj indeksa: 3123345438974
Fakultet: Pravni

Objašnjenje

Ovo je prikazano automatski zahvaljujući konstruktoru klase roditelja:

Ivica Marin
(slika)
Vinko Marin
(slika)
Marica Marin
(slika)

A ovo je zasluga samo klase nasljednice:

Godina rođenja: 1988
Broj indeksa: 45455234
Fakultet: FESB

Godina rođenja: 1980
Broj indeksa: 8978934214
Fakultet: Ekonomski

Godina rođenja: 1988
Broj indeksa: 3123345438974
Fakultet: Pravni

Operator ::

Operator dvostruko dvotočje :: omogućava dohvat konstanti, statičkih metoda i statičkih svojstava, bez instanciranja klase.

Dohvat konstante izvana bez instanciranja klase

split.php

```
<?php
class Split {
    const KONSTANTNA_VRIJEDNOST = 'Dioklecijanov palača.';
}
?>
```

split2.php

```
<?php
```

```
include "split.php";  
echo Split::KONSTANTNA_VRIJEDNOST;  
?>
```

Kad otvorimo stranicu split2.php vidjet ćemo

Dioklecijanova palača.

Objašnjenje

Ova instrukcija je izvan klase:

```
echo Split::KONSTANTNA_VRIJEDNOST;
```

Nismo instancirali objekt, a pristupili smo konstanti unutar klase.

Dohvat statičkih svojstava i statičkih metoda iznutra i izvana bez instanciranja klase

split1.php

```
<?php  
include "split.php";  
class Split1 extends Split  
{  
    public static $svojstvo = 'Početna vrijednost svojstva.';  
  
    public static function DvostrukoDvotocje() {  
        echo parent::KONSTANTNA_VRIJEDNOST . "<br />";  
        echo self::$svojstvo . "<br />";  
    }  
}  
  
Split1::DvostrukoDvotocje();  
?>
```

Na stranici će biti prikazano ovo

```
Dioklecijanova palača  
Početna vrijednost svojstva
```

Objašnjenje

split1.php

Najprije učitavamo apstraktne podatke, tj. klasu:

```
<?php  
include "split.php";
```

Pravimo klasu nasljednicu pomoću ključne riječi extends:

```
class Split1 extends Split  
{
```

Kreiramo statičko svojstvo.

Statičko svojstvo kreiramo tako što najprije pišemo ključnu riječ za vidljivost (u našem slučaju je to `public`), a potom pišemo ključnu riječ `static`. Kad pišemo nešto i na desnoj strani znaka jednakosti kao sad, to dodjeljujemo početnu vrijednost svojstvu.

```
public static $svojstvo = 'Početna vrijednost svojstva.';
```

Kreiramo statičku metodu

Najprije vidljivost a onda riječ `static`:

```
public static function DvostrukoDvotocje() {
```

Pristupamo IZNUTRA konstanti roditelja

Pristupamo iznutra konstanti roditelja pomoću ključne riječi `parent`:

```
echo parent::KONSTANTNA_VRIJEDNOST . "<br />";
```

Pristupamo iznutra svom statičkom svojstvu

Pristupamo iznutra svom statičkom svojstvu pomoću ključne riječi `self`:

```
echo self::$svojstvo . "<br />";
}
}
```

Pristupamo izvana statičkoj metodi bez instanciranja objekta

```
Split1::DvostrukoDvotocje();
?>
```

Na web stranici će biti prikazano ovo

Izvana pozvana funkcija `DvostrukoDvotocje()` će izbaciti ovo:

```
Dioklecijanov palača.
Početna vrijednost svojstva.
```

Zadaci

1. Kreiraj klasu koja ima jedno javno svojstvo i jednu javnu funkciju. Javno svojstvo ima početnu vrijednost. Javna funkcija 'printa' tu početnu vrijednost svojstva.
2. Kreiraj klasu koja ima jedno zaštićeno svojstvo i jednu funkciju. Kreiraj objekt koji printa početnu vrijednost zaštićenog svojstva.
3. Kreirajte klasu koja ima javnog člana i javnu metodu. Javni član ima neku početnu vrijednost. Preko parametra metode dodajte članu izvana vrijednost 22. Neka se na stranici prikaže vrijednost 22.
4. Što trebate dodati klasi u zadatku 3. da se na web stranici pojavi dva puta '22', ovako: 2222?
5. Što će biti prikazano na web stranici u ovom slučaju:

```
<?php
```

```

class Zadatak3
{
    public $a = 'Početna vrijednost';
    public function Prikazi($x)
    {
        print $this->a;
        print $this->a = $x;
    }
}
$objekt = new Zadatak3();
echo $objekt->Prikazi("22");
?>

```

6. a što u ovom slučaju:

```

<?php
class Zadatak3
{
    public $a = 'Početna vrijednost';
    public function Prikazi($x)
    {
        print $this->a = $x;
        print $this->a;
    }
}
$objekt = new Zadatak3();
echo $objekt->Prikazi("22");
?>

```

7. Napravite klasu koja ima nekoliko privatnih članova i dvije funkcije. Pomoću jedne funkcije se dodaju vrijednosti tim članovima izvana, a pomoću druge funkcije se dohvaćaju vrijednosti tih članova izvana. Stavite članovima i početne vrijednosti.

Napomena: ključna riječ return vraća vrijednost člana:

```

public function DohvatiVrijednostB()
{
    return $this->b;
}

```

- Ako ne dodijelite izvana nove vrijednosti, koje će vrijednosti imati članovi?
 - Ako dodijelite izvana nove vrijednosti, koje će vrijednosti imati članovi, početne ili novo dodijeljene?
- Napravite formu pomoću koje ćete dodjeljivati vrijednosti članovima u zadatku 7.
 - Napravi klasu koja ima konstruktor.
 - Napravi klasu kojoj se vrijednosti konstruktora dodaju izvana preko parametara klase.
 - Napravi nekoliko primjera klasa s operatorom :: i ključnim riječima static, parent, self, const.
 - Napravi klasu MojaSkola.

13. Napravi klasu MojKucniLjubimac.
14. Napravi klasu MojRadniDan.
15. Napravi klasu MojBlog.
16. Posjetite www.php.net

Rješenje zadataka

1. zadatak

```
<?php
class MojaPrvaKlasa
{
    public $javnaVarijabla = 'Javna varijabla';
    public function TiskajVarijable()
    {
        print $this->javnaVarijabla;
    }
}
$objekt = new MojaPrvaKlasa();
echo $objekt->javnaVarijabla;
?>
```

2. zadatak

```
<?php
class MojaPrvaKlasa
{
    protected $zasticenaVarijabla = 'Zaštićena varijabla';
    public function TiskajVarijable()
    {
        print $this->zasticenaVarijabla;
    }
}
$objekt = new MojaPrvaKlasa();
echo $objekt->TiskajVarijable();
?>
```

3. zadatak

```
<?php
class Zadatak3
{
    public $a = 'Početna vrijednost';
    public function Prikazi($x)
    {
        print $this->a = $x;
    }
}
$objekt = new Zadatak3();
echo $objekt->Prikazi("22");
?>
```

4. zadatak

```
<?php
class Zadatak3
{
```

```

public $a = 'Početna vrijednost';
public function Prikazi($x)
{
    print $this->a = $x;
    print $this->a;
}
}
$objekt = new Zadatak3();
echo $objekt->Prikazi("22");
?>

```

5. zadatak

Početna vrijednost22

6. zadatak

2222

7. zadatak

```

<?php
class MojaPrvaKlasa
{
    private $a = "A";
    private $b = "B";

    public function DodijeliVrijednost($x, $y)
    {
        $this->a = $x;
        $this->b = $y;
    }
    public function DohvatiVrijednostA()
    {
        return $this->a;
    }
    public function DohvatiVrijednostB()
    {
        return $this->b;
    }
}

$objekt = new MojaPrvaKlasa();
echo $objekt->DohvatiVrijednostB();
?>

```

a. Na ekranu će biti 'B'.

```

$objekt = new MojaPrvaKlasa();
$objekt->DodijeliVrijednost(12, 24);
echo $objekt->DohvatiVrijednostB();
?>

```

b. Na ekranu će biti '24'.

8. zadatak

Stranica zadatak8a.php

```

<?php
class MojaPrvaKlasa

```



```

{
    private $a = "A";
    private $b = "B";

    public function DodijeliVrijednost($x, $y)
    {
        $this->a = $x;
        $this->b = $y;
    }
    public function DohvatiVrijednostA()
    {
        return $this->a;
    }
    public function DohvatiVrijednostB()
    {
        return $this->b;
    }
}

$objekt = new MojaPrvaKlasa();
$objekt->DodijeliVrijednost($_GET[x], $_GET[y]);
echo $objekt->DohvatiVrijednostB();
?>

```

Stranica zadatak8b.php

```

<form action='zadatak8a.php' method='GET'>
<input name='x' type='text' />
<input name='y' type='text' />
<input type='submit' value="Pohrani" />
</form>

```

Test

1. Što će izbaciti ova stranica:

```

<?php
class MojaPrvaKlasa
{
    public $javnaVarijabla = 'Javna varijabla';
    protected $zasticenaVarijabla = 'Zaštićena varijabla';
    private $privatnaVarijabla = 'Privatna varijabla';

    public function TiskajVarijable()
    {
        echo $this->zasticenaVarijabla;
        echo $this->privatnaVarijabla;
    }
}

$objekt = new MojaPrvaKlasa();
echo $objekt->zasticenaVarijabla;
echo $objekt->privatnaVarijabla;
echo $objekt->javnaVarijabla;
?>

```

2. Koju ulogu igra \$this?

3. Koju ulogu igra ->?
4. Čemu služi operator dvostruko dvotočje ::?
5. Čemu služi self ::?
6. Čemu služi parent ::?

Odgovori na test

1. Ništa. Zato što se nakon instanciranja objekta izvodi instrukcija koja proizvodi grešku i program se na njoj zaustavlja:
`echo $objekt->zasticenaVarijabla;`
2. \$this znači 'ja' i koristi se za pristup svojstvima i funkcijama unutar klase.
3. Strelica -> je isto što i točka u ASP.NET. Služi za pozivanje svojstava i funkcija.
4. Za dohvat konstanta, statičkih članova i statičkih metoda izvana, bez kreiranja instance objekta.
5. self :: služi za dohvat statičkog člana, statičke metode i konstante iznutra.
6. Dijete pomoću parent :: dohvaća iznutra statičke članove, statičke metode i konstante roditelja.



PHP Smarty

Metodika-----

Smarty je paket PHP klasa pomoću kojih ćemo razdvojiti XHTML od PHP-a. U ovoj nastavnoj jedinici učimo koristiti Smarty od nule.

Što se tu događa?

Do sad smo naučili da je XHTML dokument taj u kojem pišemo PHP kôd. Takva web stranica na serveru je uvijek XHTML i PHP. Međutim, takva stranica nikad ne dolazi kompletna u Web preglednik. Dolazi drukčija nego je na serveru. Dolazi SAMO XHTML a umjesto PHP kôda dolazi REZULTAT procesuiranja tog PHP kôda. Taj rezultat je SADRŽAJ. Najčešće iz baze podataka.

Tko je napravio php web stranicu? Napravili su je programer i grafički web dizajner. Najprije je radio jedan pa proslijedio stranicu drugom da on ubaci svoj dio. Onda treba nešto mijenjati onaj prvi, pa drugi, pa opet prvi i sve tako do finalnog proizvoda. Kad se tako radi imamo zastoje radi:

1. Čekaju jedan drugoga.
2. Slučajno dizajner obriše PHP kôd i obrnuto, programer XHTML.
3. Teže se snaći i jednom i drugom u izvornom kôdu u kojem su izmješani PHP i XHTML.

Autori Smartya su imali gornje zastoje pred očima kad su odlučili napraviti PHP klase koje će omogućiti razdvajanje web stranica u dva dokumenta. Jedan će biti čisti XHTML a drugi čisti PHP.

PHP dokument će izvući SADRŽAJ iz baze podataka i pohraniti ga u JAVNE varijable. Tad nastupaju Smarty objekti. Oni uzimaju te varijable iz RAM-a i raspoređuju ih u XHTML dokument.

Udžbenik-----

Što ćemo naučiti?

1. Kako Smarty razdvaja XHTML od PHP.
2. index.tpl i index.php.
3. Smarty i podaci iz baze podataka.
4. Smarty i ARRAY.
5. Smarty i ASOCIJATIVNI ARRAY.

Razdvajanje XHTML od PHP

Instalacija vježbi

U kompletu datoteka koje idu uz ovaj obrazovni sadržaj je i mapa 'Poglavlje_smarty'. Kopirajte tu mapu na svoje računalo, u mapu 'htdocs' ili drugu koju čita Apache na vašem računalu. Mapa ima nekoliko podmapa imenovanih s1, s2 itd. Ukoliko želite pogledati gotove primjere, morate prethodno otvoriti index.php svakog primjera i promijeniti navedene fizičke adrese mapa, sukladno vašoj situaciji.

Cilj vježbi

U svim vježbama, kad otvorimo index.php, dobit ćemo najjednostavniji mogući sadržaj:

```
Dobro nam se vratili, Ivica Kartelo!
```

Uzeli smo najjednostavniji mogući primjer, kako bi sva naša pozornost bila usmjerena na elemente smartya.

Izvorni kôd na klijentu će uvijek biti isti:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../stil.css" type="text/css" />
</head>
<body>
<h1>Dobro nam se vratili, Ivica Kartelo!</h1>
</body>
</html>
```

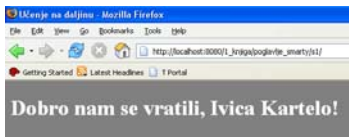
Međutim, ono što ćete imati na serveru, jako će se razlikovati. Do sad nam je uvijek bila potrebna samo jedna stranica i na serveru. Namjerno ćemo koristiti PHP za ovaj jednostavni primjer, kako bismo što bolje naučili kako smarty razdvaja XHTML od PHP. Kad želimo pomoću smartya razdvojiti PHP od XHTML, potrebne su bar dvije stranice na serveru: index.php i index.tpl i nekoliko mapa. Idemo redom.

Najprije se podsjetimo kako smo radili bez smartya.

Vježba 1

XHTML i PHP zajedno u istom dokumentu

1. Otvorite podmapu s1/index.php i vidjet ćete (kod mene url izgleda ovako: http://localhost:8080/1_knjiga/poglavlje_smarty/s1/)



2. Izvorni kôd je XHTML dokument u kojem je napisan i PHP kôd:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../../stil.css" type="text/css" />
</head>
<body>
<h1>
<?php
$ime = "Ivica Kartelo";
echo "Dobro nam se vratili, " . $ime . "!";
?>
</h1>
</body>
</html>
```

Istakao sam php kôd. Tako smo radili u svim dosadašnjim primjerima u knjizi. PHP je 'otok' u XHTML-u. U sljedećem primjeru ćemo vidjeti kako Smarty razdvaja XHTML od PHP.

Vježba 2

XHTML i PHP razdvojeni

Napomena: u mapi s2 je mapa biblioteka. Mapu biblioteka kopirajte u sve ostale mape s3, s4 itd. ukoliko je tamo nema. U mapi biblioteka je klasa SMARTY, a smarty nam treba u svim sljedećim primjerima.

Smarty je skupina php klasa. Smarty je otvoreni kôd i skida se s Interneta. Za sve primjere u ovoj knjizi, možete koristiti smarty isporučen (download s www.e92.hr) uz ovu knjigu.

1. Kreirajte mapu 'a' koju čita Apache (u mapi 'htdocs'). To će biti korijen mapa naše prve smarty aplikacije.

Alias

Ili još bolje, napravite alias mape 'a' ovako:

2. Napravite mapu 'a' za vaše vježbe na primjer ovdje C:\a\
3. Otvorite konfiguracijsku datoteku od Apachea. To je datoteka httpd.conf i na mom računalu se nalazi ovdje:

```
C:\Program Files\Apache Group\Apache2\conf\
```

4. U sekciji aliases:

```
# Aliases: Add here as many aliases as you need (with no
limit). The format is
# Alias fakename realname
```

5. tipkajte sljedeći istaknuti tekst:

```
# Aliases: Add here as many aliases as you need (with no
limit). The format is
# Alias fakename realname
```

```
Alias /a/ "c:/a/"
Alias /a/ "c:/a"
```

6. Pohranite datoteku httpd.conf

7. Restartajte Apache.

Podsjetnik: Start/Control Panel/Administrative Tools/Services/
Pronađete servis Apache, kod mene je to servis Apache2, označite ga, pa u
lijevom gornjem kutu kliknite na Restart.

8. U mapi 'a' napravite ove podmape: biblioteke, cache, configs, templates, templates_c.

9. U mapu 'biblioteke' kopirajte mapu 'smarty' iz s1/biblioteke/smarty/. Tako imate sve potrebne klase smarty.

10. U mapi templates napravite novi dokument u Notepadu, imenujte ga index.tpl i u njega tipkajte ovaj XHTML kôd:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../../../stil.css" type="text/css" />
</head>
<body>
<h1>Dobro nam se vratili, {$ime}</h1>
</body>
</html>
```

NOVOST: Jedina novost u ovom XHTML dokumentu je **{*\$ime*}**. To je varijabla koja se zove *\$ime* i koju će koristiti klase u mapi smarty. Napisana je u vitičastim zagradama. Cijelu sintaksu te varijable, zajedno s vitičastim zagradama ste već sreli u ovoj knjizi. To je uobičajena sintaksa php.

11. U mapi 'a', korijen mapi naše tekuće aplikacije, kreirajte Notepad dokument index.php:

```
<?php

require('C:/a/biblioteke/Smarty/Smarty.class.php');
$smarty = new Smarty();
```

```

$smarty->template_dir = 'C:/a/templates';
$smarty->compile_dir = 'C:/a/templates_c';
$smarty->cache_dir = 'C:/a/cache';
$smarty->config_dir = 'C:/a/configs';

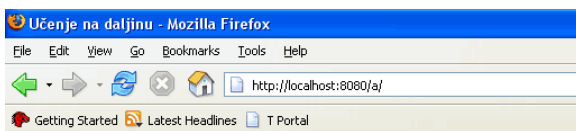
$smarty->assign('ime', 'Ivica Kartelo');
$smarty->display('index.tpl');

?>

```

Obratite pozornost na istaknuti dio php kôda: ime i index.tpl. Varijabla koja je u dokumentu index.tpl navedena u ovoj sintaksi {\$ime}, sad je napisana 'ime' i argument je funkcije assign().

12. U web pregledniku otvorite index.php i dobit ćete ovu web stranicu:



Dobro nam se vratili, Ivica Kartelo!

Objašnjenje

templates_c

13. Pogledajte u mapu templates_c. Više nije prazna. Tijekom prvog učitavanja u web preglednik, klasa smarty kreira web stranicu index.tpl.php i spremi je u ovu mapu. To je predložak kojeg više ne mora praviti, pa su sljedeća učitavanja brža. Ukoliko bilo što promijenimo na stranicama, smarty će kreirati i pohraniti u ovu mapu osvježeni predložak. Autori Smartya su to nazvali kompajliranje predložaka. Kompajliranje je kreacija binarne datoteke, što ovdje nije slučaj. Predložak je php skript, a datoteka index.tpl.php je tekstualna, a ne binarna.

index.php

To je čisti PHP kôd! Pa gdje mu je XHTML? Čitajte dalje. U ovom PHP kôdu je učinjeno sljedeće.

```
<?php
```

Pomoću funkcije require učitava se datoteka Smarty.class.php. Vi ćete navesti apsolutni put datoteke kakav je na vašem računalu. Tako se učitala klasa Smarty.

```
require('C:/a/biblioteke/Smarty/Smarty.class.php');
```

Instanciramo objekt \$smarty. Tako apstraktna klasa Smarty dobija svoj konkretni objekt.

```
$smarty = new Smarty();
```


Kreiramo svojstva objekta \$smarty: template_dir, compile_dir, cache_dir, config_dir. Tim svojstvima dodijeljujemo apsolutne fizičke adrese odgovarajućih mapa: templates, templates_c, cache, configs.

```
$smarty->template_dir = 'C:/a/templates';  
$smarty->compile_dir = 'C:/a/templates_c';  
$smarty->cache_dir = 'C:/a/cache';  
$smarty->config_dir = 'C:/a/configs';
```

Ovaj kôd je najzanimljiviji. Objekt \$smarty poziva funkciju assign i dodijeljuje joj dva argumenta: 'ime' i 'Ivica Kartelo'. Prvi argument je varijabla za smarty predložak, a drugi argument je vrijednost te varijable.

```
$smarty->assign('ime', 'Ivica Kartelo');
```

Naš objekt poziva funkciju display() koja ima samo jedan argument, a to je smarty index.tpl. Funkcija display() će kreirati našu Home Page.

```
$smarty->display('index.tpl');
```

```
?>
```

index.tpl

Sve mape koje vidite u mapi s2 su potrebne i zato ste ih napravili u 'a'. Sve ćemo ih postepeno upoznati. Do sad znamo su klase smarty u mapi 'biblioteka'. Sad znamo da je u mapi 'templates' datoteka index.tpl – smarty predložak.

Datoteke index.php i index.tpl su dvije datoteke koje su razdvojile XHTML od PHP. PHP klase koje se nalaze u mapi smarty, prilikom procesuiranja index.php, u izlaznu datoteku ugrađuju i XHTML kôd iz index.tpl. Datoteka index.tpl je smarty predložak.

Jedina novost u XHTML-u index.tpl je ovaj redak, i u njemu varijabla \$ime u vitičastim zagradama:

```
<h1>Dobro nam se vratili, {$ime}!</h1>
```

Ključne riječi assign i display

Obratite pozornost na ključne riječi **assign** i **display**. Uz prvu je vezana varijabla predloška i sadržaj predloška, a uz drugu je vezan predložak index.tpl. Dalje u tekstu ćete naučiti, kako sadržaj 'Ivica Kartelo' može doći i iz baze podataka, što i je najčešći slučaj.

Zaključak

U index.tpl se nalaze SAMO varijable. U index.php se nalaze funkcije ASSIGN() i DISPLAY(). Funkcija ASSIGN() ima dva argumenta. Prvi je VARIJABLA a drugi je SADRŽAJ. U nastavku ćemo vidjeti kako izgleda kôd kad SADRŽAJ uzimamo iz baze.

Vježba 3

Sadržaj iz baze podataka bez smartya

Najprije se prisjetimo čitanja sadržaja iz baze na dosadašnji način, bez smartya. Trebat će vam baza 'posjetitelji'.

1. Napišite u Notepadu sljedeći SQL kôd za kreiranje baze 'posjetitelji':

```
DROP DATABASE IF EXISTS `posjetitelji`;

CREATE DATABASE posjetitelji;

USE posjetitelji;

DROP TABLE IF EXISTS `imena`;

CREATE TABLE imena (
    ime varchar(50) NOT NULL default ''
) TYPE=MyISAM;
INSERT INTO imena (ime) values ('Ivica Kartelo');
```

Napomena: U mapi s3 imate i datoteku 'posjetitelji.sql' potrebnu za kreiranje baze.

2. Pohranite u mapu 'bin' (na mom računalu: C:\Program Files\MySQL\MySQL Server 4.1\bin). Otvorite u Command Prompu datoteku bin>. Tipkajte na naredbodavnom retku bin>

```
bin>mysql -u root < posjetitelji.sql
Enter
```

1. PHP stranica pomoću koje smo prikazivali sadržaj iz baze, izgledala je ovako:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../../../stil.css" type="text/css" />
</head>
<body>
<h1>

<?php
```

```
mysql_connect("localhost", "localhost@root", "ivica");
mysql_select_db("posjetitelji");
$result = mysql_query("SELECT ime FROM imena;");
while ($row = mysql_fetch_assoc($result)) {
    extract($row, EXTR_PREFIX_ALL, "petlja");
    echo "Dobro nam se vratili, " . $petlja_ime .
```

```
"!";
```

```

    }
    ?>
</h1>
</body>
</html>

```

Iz prijašnjih vježbi u ovoj knjizi smo naučili kako radi gornji kôd. Pogledajmo sad kako će ovu stranicu XHTML-a i PHP-a razdvojiti Smarty.

Vježba 4

Podaci iz baze podataka prikazani pomoću Smartya

Mapa s4. Cijela vježba je u toj mapi. Ukoliko ćete kopirati mapu s4 na svoje računalo, morate promijeniti fizičke adrese mapa navedenih u index.php.

1. Vježbu 2 premjestite u neku drugu mapu, kako biste oslobodili mapu C:\a\ za ovu vježbu.
2. U mapi 'a' napravite ove podmape: biblioteke, cache, configs, templates, templates_c.
3. U mapu 'biblioteke' kopirajte mapu 'smarty' iz s1/biblioteke/smarty/. Tako imate sve potrebne klase smarty.

index.tpl

4. U mapi templates napravite novi dokument u Notepadu, imenujte ga index.tpl i u njega tipkajte ovaj XHTML kôd, koji se ne razlikuje od onog koda u vježbi 2.

```

<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../stil.css" type="text/css" />
</head>
<body>
<h1>Dobro nam se vratili, {$ime}</h1>
</body>
</html>

```

index.php

5. U mapi 'a' napravite index.php:

```

<?php
//Put do klase Smarty
require('C:/1_knjiga/Poglavlje_smarty/s4/biblioteke/Smarty/Sm
arty.class.php');

mysql_connect("localhost", "localhost@root", "ivica");

```

```
mysql_select_db("posjetitelji");
$result = mysql_query("SELECT ime FROM imena;");
while ($row = mysql_fetch_assoc($result)) {
    extract($row, EXTR_PREFIX_ALL, "petlja");
    $ime = $petlja_ime;
}

$smarty = new Smarty();

$smarty->template_dir = 'C:/a/templates';
$smarty->compile_dir = 'C:/a/templates_c';
$smarty->cache_dir = 'C:/a/cache';
$smarty->config_dir = 'C:/a/configs';
$smarty->assign('ime', $ime);
$smarty->display('index.tpl');
?>
```

Objašnjenje

Na stranici index.php ponovo imamo čisti PHP kôd.

Ističem samo važni dio kôda. U ovom retku je kreirana varijabla \$ime i odmah joj je dodijeljena vrijednost onog što je upit donio iz baze ('Ivica Kartelo').

```
$ime = $petlja_ime;
```

U ovom retku je 'ime' varijabla smarty predloška, a '\$ime' je sadržaj koji je došao iz baze ('Ivica Kartelo'):

```
$smarty->assign('ime', $ime);
```

Funkcija display pozove smarty predložak index.tpl na kojem će se dogoditi assign('ime', \$ime), dodjeljivanje sadržaja varijabli.

```
$smarty->display('index.tpl');
```

templates_c

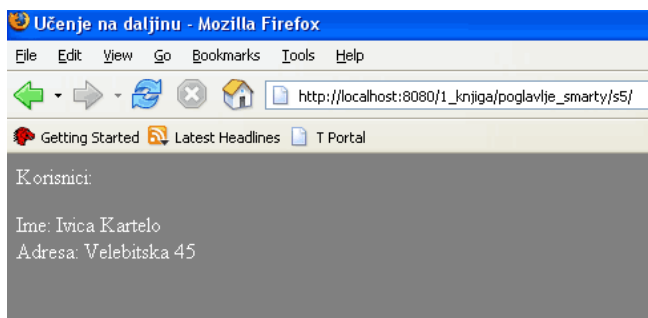
U ovoj mapi je klasa Smarty kreirala ('kompajlira') predložak index.tpl.php. Tako ne treba prilikom svakog učitavanja kreirati taj predložak, pa je učitavanje brže.

Vježba 5

Vježba se nalazi u mapi s5.

Više sadržaja

U ovoj vježbi imamo više podataka na web stranici:



Mapa s5. Cijela vježba je u toj mapi. Ukoliko ćete kopirati mapu s5 na svoje računalo, morate promijeniti fizičke adrese mapa navedenih u index.php.

6. Vježbu 4 premjestite u neku drugu mapu, kako biste oslobodili mapu C:\a\ za ovu vježbu.
7. U mapi 'a' napravite ove podmape: biblioteke, cache, configs, templates, templates_c.
8. U mapu 'biblioteke' kopirajte mapu 'smarty' iz s1/biblioteke/smarty/. Tako imate sve potrebne klase smarty.
9. Kreirajte index.tpl u mapi 'templates' i index.php u mapi 'a'.

index.tpl

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../../stil.css" type="text/css" />
</head>
<body>
Korisnici:<p>

Ime: {$ime}<br>
Adresa: {$adresa}<br>

</body>
</html>
```

index.php

```
<?php

// Apsolutni put datoteke Smarty.class.php.
require('C:/1_knjiga/Poglavlje_smarty/s5/biblioteke/Smarty/Smarty.class.php');

//Kreiranje objekta $smarty.
$smarty = new Smarty();
```

```
//Kreiranje potrebnih konstanti unutar objekta &smarty
//koje će objektu $smarty reći put do smarty-mapa.
$smarty->template_dir = 'C:/a/templates';
$smarty->compile_dir = 'C:/a/templates_c';
$smarty->cache_dir = 'C:/a/cache';
$smarty->config_dir = 'C:/a/configs';

//Dodavanje (assign) sadržaja.Ovaj sadržaj najčešće dolazi iz
baze ili drugih izvora.
//Mi ćemo u ovom primjeru koristiti statički sadržaj.
$smarty->assign('ime', 'Ivica Kartelo');
$smarty->assign('adresa', 'Velebitska 45');

//Neka objekt $smarty prikaže predložak.
$smarty->display('index.tpl');

?>
```

Objašnjenje

U gornjem kôdu imamo dvije instrukcije s funkcijom assign:

```
$smarty->assign('ime', 'Ivica Kartelo');
$smarty->assign('adresa', 'Velebitska 45');

//Neka objekt $smarty prikaže predložak.
$smarty->display('index.tpl');
```

U smarty predlošku **index.tpl** ćemo imati dvije varijable: ime i adresa:

```
Ime: {$ime}<br>
Adresa: {$adresa}<br>
```

Vježba 6

Ovaj primjer ima za cilj pokazati kako se smarty predložak index.tpl može podijeliti u više predložaka: index.tpl, zaglavlje.tpl, podnožje.tpl, te kako možemo vršiti formatiranje datuma.

zaglavlje.tpl

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Učenje na daljinu</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../stil.css" type="text/css" />
</head>
<body>
```

index.tpl

```
{include file="zaglavlje.tpl"}

{$smarty.now|date_format:"%d.%m.%Y"}<p>
Korisnici:</p>
<p>
Ime: {$ime}<br />
Adresa: {$adresa}<br />
</p>

{include file="podnozje.tpl"}
```

podnozje.tpl

```
</body>
</html>
```

index.php

Ova stranica je ista kao u Vježbi 5.

templates_c

U ovoj mapi će biti 'kompajlirana' tri predloška: zaglavlje.tpl, index.tpl, podnozje.tpl.

Objašnjenje index.tpl

Poziva se pomoću include zaglavlje.tpl. Sintaksa je različita od one u PHP:

```
{include file="zaglavlje.tpl"}
```

I ovdje funkcija NOW znači današnji datum. Vidimo kako se poziva u smartyu i kako se formatira.

```
{{$smarty.now|date_format:"%d.%m.%Y"}}<p>
Korisnici:</p>
<p>
```

Raspoređujemo varijable \$ime i \$adresa po želji u XHTML:

```
Ime: {$ime}<br />
Adresa: {$adresa}<br />
</p>
```

I na kraju pozivamo podnozje.tpl:

```
{include file="podnozje.tpl"}
```

Vježba 7

array

index.php

Izdvajamo samo kôd po kojem se ovaj primjer razlikuje od prethodnog:

```
$smarty->assign('id', array(1,2,3,4,5));  
$smarty->assign('imena',  
array('ivo','mate','vinko','marta','iva'));  
  
$smarty->display('index.tpl');
```

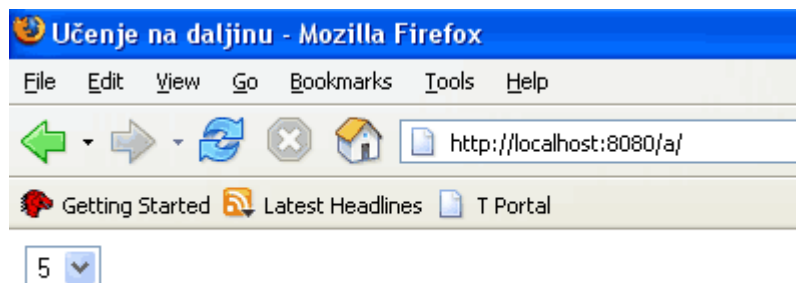
Varijabli id dodjeljuje se jedan array, varijabli imena drugi array. U realnim projektima ti arrayi dolaze iz baze.

index.tpl

```
{include file="zaglavlje.tpl"}  
  
<select>  
{html_options values=neko_ime output=$id selected="5"}  
</select>  
  
{include file="podnozje.tpl"}
```

Kreiran je XHTML element <SELECT> od **id**.

Web stranica će prikazati ovo



Vježba 8

Primjer u mapi s8.

1. Zadržite sadržaj Vježbe 7. u mapi C:\a\
2. Promijenite samo stranice index.php i index.tpl, kao u nastavku.

Asocijativni array

index.php

```
<?php

// Apsolutni put datoteke Smarty.class.php.
require('C:/a/biblioteke/Smarty/Smarty.class.php');

// Kreiranje objekta $smarty.
$smarty = new Smarty();

//Kreiranje potrebnih konstanti unutar objekta &smarty
//koje će objektu $smarty reći put do smarty-mapa.
$smarty->template_dir = 'C:/a/templates';
$smarty->compile_dir = 'C:/a/templates_c';
$smarty->cache_dir = 'C:/a/cache';
$smarty->config_dir = 'C:/a/configs';

//Neka sadržaj bude array.
$smarty->assign('ime',
array('Marta','Vinko','Iva','Mate','Ivana'));

//Neka sadržaj bude asocijativni array.
$smarty->assign('korisnici', array(
    array('ime' => 'Marta', 'tel' => '555-3425'),
    array('ime' => 'Vinko', 'tel' => '555-4364'),
    array('ime' => 'Iva', 'tel' => '555-3422'),
    array('ime' => 'Mate', 'tel' => '555-4973'),
    array('ime' => 'Ivana', 'tel' => '555-3235')
));

//Neka objekt $smarty prikaže predložak.
$smarty->display('index.tpl');
```

index.tpl

```
{include file="zaglavlje.tpl"}
<table>
{section name=neko_ime loop=$ime}
{strip}
    <tr bgcolor="{cycle values="#gggggg,#ffffcc"}">
        <td>{$ime[neko_ime]}</td>
    </tr>
{/strip}
{/section}
</table>

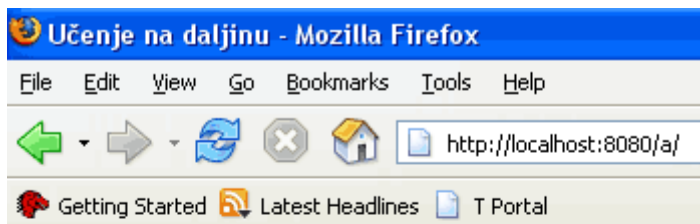
<table>
{section name=petlja loop=$korisnici}
{strip}
    <tr bgcolor="{cycle values="gray,gold"}">
        <td>{$korisnici[petlja].ime}</td>
        <td>{$korisnici[petlja].tel}</td>
    </tr>
```

```

{/strip}
{/section}
</table>
{include file="podnozje.tpl"}

```

Web stranica



Marta

Vinko

Iva

Mate

Ivana

Marta 555-3425

Vinko 555-4364

Iva 555-3422

Mate 555-4973

Ivana 555-3235

Objašnjenje

Asocijativni array uvijek izgleda ovako:

```

array(
    array('ime' => 'Marta', 'tel' => '555-3425'),
    array('ime' => 'Vinko', 'tel' => '555-4364'),
    array('ime' => 'Iva', 'tel' => '555-3422'),
    array('ime' => 'Mate', 'tel' => '555-4973'),
    array('ime' => 'Ivana', 'tel' => '555-3235')
)

```

Takav array nam najčešće dolazi iz baze s upitom i iz takvog arraya nam je najlakše izvaditi podatak. Zašto? Zato što je u asocijativnom arrayu uz svaki podatak pridruženo i ime stupca tablice. Zato se i zove asocijativan ili pridružen. Kasnije u aplikacijama, a i sad ovdje vidimo koja je velika korist od asocijativnog arraya. Ne moramo pozivati vrijednosti po brojevima, na primjer `array[0][0]= Marta`, nego pomoću imena pridruženog stupca:

```
ime[0]=Marta
```

```
ime[1]
```

```
ime[2]
ime[3]
ime[4]
tel[0]
tel[1]
tel[2]
tel[3]
tel[4]
```

Petlje koje u smartyevoj sintaksi izvlače vrijednosti iz asocijativnih arraya i raspoređuje ih u HTML tablice, izgledaju ovako, za varijablu \$ime:

```
{section name=mysec loop=$ime}
{strip}
  <tr bgcolor="{cycle values="#ggggggg,#000000"}">
    <td>{$ime[mysec]}</td>
  </tr>
{/strip}
{/section}
```

i varijablu \$korisnici:

```
{section name=petlja loop=$korisnici}
{strip}
  <tr bgcolor="{cycle values="#aaaaaa,#bbbbbb"}">
    <td>{$korisnici[petlja].ime}</td>
    <td>{$korisnici[petlja].tel}</td>
  </tr>
{/strip}
{/section}
```

Ovdje je zanimljivo i formatiranje redaka pomoću smarty ključne riječi cycle. Jedan redak u boji #ggggggg, a sljedeći u boji #ffffcc i sve tako naizmjenice.

```
<tr bgcolor="{cycle values="#ggggggg,#ffffcc"}">
<tr bgcolor="{cycle values="gray,gold"}">
```

Zaključak

Iz gornjih primjera razdvajanja XHTML od PHP pomoću klasa Smarty možemo zaključiti sljedeće.

1. index.tpl je XHTML i pohranjuje se u mapu 'templates',
2. index.php je PHP i pohranjen je u korijen mapu,
3. index.tpl sadrži varijable smarty predloška,
4. index.php sadrži funkciju assign() koja za argumente ima varijablu predloška smarty i sadržaj te varijable,
5. index.php sadrži funkciju display() koja za argument ima index.tpl,
6. sadržaj može biti kratki string, array ili asocijativni array.

U praksi najčešće kao sadržaj dolazi asocijativni array iz baze podataka i upravo je to naša sljedeća nastavna cjelina. Za primjer uzimam Knjigu gostiju. Osim asocijativnog arraya iz baze, ovdje se upoznajemo i s tro slojnom filozofijom projektiranja web aplikacija. Također upoznajemo i kako se web aplikacija spaja na bazu podataka pomoću PEAR klasa.

Nastavna cjelina 8



PHP SMARTY I PEAR

Metodika-----

Ono što 'leži' na serveru od naše početne stranice **index.php**:

```
<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>
```

A ovako će ta stranica izgledati u našem Web pregledniku:



vrh_aplikacije.php

```
<?php
require_once 'konfiguracija.inc.php';
require_once 'uspostavljanje_smarty.php';
require_once 'bazapodataka.php';

//Instanciranje globalnog objekta BazaPodataka
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
?>
```

Na početku kôda cijele aplikacije instancirali smo objekt:

```
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
```

Posjetimo se Nastavne jedinice 1. Varijabla \$x:

```
<?php
```

```
$x = 1;
?>
```

je GLOBALNA varijabla. Svaki naš PHP program automatski stvara array \$GLOBALS i u njega sprema sve globalne varijable. Da bi naši veliki programi bili što jasniji, ispred takvih varijabli dodajemo slovo 'g':

```
<?php
$gX = 1;
?>
```

Ista stvar se događa sad u OOP. Instanciranje objekta je isto što i kreiranje varijable. Na lijevoj strani znaka jednako, pišemo ime varijable sa znakom dolara ispred i slovom 'g', a na desnoj strani pišemo new ImeKlase(parametar klase). Kažemo ovako: instancirali smo novi objekt i dodijelili ga varijabli `toj` i `toj`.

```
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
```

Instancirali smo novi objekt `BazaPodataka` i dodijelili ga varijabli `$gBazaPodataka`.

U Nastavnoj jedinici 1 smo naučili da su varijable unutar funkcija LOKALNE. Ako unutar funkcije želimo dohvatiti GLOBALNU varijablu, pišemo ovako \$GLOBALS['gBazaPodataka']:

```
function __construct()
{
    $this->bPodataka = $GLOBALS['gBazaPodataka'];
}
```

Pomoću tog objekta ostvarili smo spoj naše aplikacije na bazu podataka. Taj spoj je resurs kojeg smo pohranili u globalnu varijablu `$gBazaPodataka`. Globalna varijabla se naziva globalna zato što živi koliko god i program. Svi PHP dokumenti koje uključujemo u aplikaciju pomoću INCLUDE također imaju pristup globalnoj varijabli. Ti dokumenti su sastavni dio kôda aplikacije baš kao da su instrukcije direktno ispisane u `index.php`. Mapu INCLUDE smo kreirali samo zato da bismo u nju pohranili zasebne klase koje možemo INCLUDE u aplikaciju `index.php`. Tako su nam klase sve u jednoj mapi, na zasebnim fizičkim datotekama i možemo ih kao takve uvijek koristiti u drugim našim aplikacijama. To je dobra organizacija i posla i aplikacije.

dno_aplikacije.php

```
<?php
$gBazaPodataka->BpDisconnect();
?>
```

Možemo pisati i ovako:

```
<?php
$GLOBALS['gBazaPodataka']->BpDisconnect();
?>
```

ali nije potrebno, jer nije UNUTAR FUNKCIJE.

Na dnu aplikacije smo eksplicitno prekinuli vezu s bazom i tako je MySQL baza rasterećena od jednog korisnika.

Ovdje je naglasak na

Smarty

Pear

Aplikacija u tri sloja

Smarty

Uvijek se treba vraćati na prethodno poglavlje da se dobro uhvati način kôdiranja Smarty predložaka. Ovdje je korak dalje. Smarty dobija podatke iz baze. Dobija asocijativne arraye. Podatke iz baze ugrađuje u prezentacijski sloj – web stranice. Primjer Knjiga gostiju nam samo služi da na postepen način naučimo kako podaci iz baze dolaze Smartyu i kako ih ovaj ugrađuje u predloške web stranica. Naučeno ovdje primjenit ćemo mnogo puta u sljedeća dva poglavlja: CMS i E-trgovina. Zato je važno to ovdje dobro naučiti i uvijek se ovdje vraćati ako treba utvrditi gradivo. Ovdje učimo Smarty, a u CMS to ponavljamo.

Pear

Pear je najčešći php paket kojeg danas programeri koriste za spajanje na bazu podataka. Bilo koju, ne samo MySQL. Ovdje je prilika detaljno i postepeno naučiti kako PEAR to čini. I onda to koristiti Copy/Paste u našim budućim aplikacijama, CMS i E-trgovina. Ovdje to treba naučiti jer ima malo kôda baš radi koncentracije na Smarty, PEAR i troslojnu konfiguraciju.

Aplikacija u tri sloja

Današnje web aplikacije se programiraju u tri sloja: prezentacijski, poslovni i podatkovni. Prezentacijski sloj su web stranice. Taj sloj služi za interakciju čovjeka i web aplikacije. Taj sloj je grafičko sučelje, GUI, aplikacije.

Podatkovni sloj su SQL upiti. Taj sloj komunicira s bazom podataka. SQL smo učili radi tog sloja. Taj sloj pomoću SQL upita izvlači tablice, podatke, briše podatke, ažurira podatke, dodaje nove retke podataka u tablice.

Poslovni sloj je sloj IZMEĐU prezentacijskog i podatkovnog sloja. Preko njega se odvija protok SVIH informacija unutar aplikacije. Ne može prezentacijski sloj doći u vezu s podatkovnim slojem direktno. Samo preko poslovnog sloja. I obrnuto. Posjetitelj ispuni formu na web stranici i klikne na gumb Šalji. To se događa na prezentacijskom sloju. Prezentacijski sloj prihvaća te podatke i kao parametre ih šalje poslovnom sloju. Poslovni sloj ih prosljeđuje podatkovnom sloju koji ih pohrani u bazu.

Da li poslovni sloj služi samo za prosljeđivanje podataka u jednom ili drugom smjeru? Ne samo za to. U našim aplikacijama u ovoj knjizi uglavnom za to, osim kad je tražilica u pitanju i kad smo više posla dali poslovnom sloju. U poslovni sloj možemo ugrađivati po želji poslovna pravila firme, a da ne moramo intervenirati, bilo što mijenjati u ostala dva sloja.

U ovoj nastavnoj jedinici i prethodnoj su temelji naših velikih aplikacija CMS i E-trgovina. Zato ovdje treba dobro proučiti rad sa Smarty, PEAR i tro slojnom konfiguracijom.

Udžbenik-----

Što ćemo naučiti?

1. Kako PEAR spaja aplikaciju na bazu podataka.
2. Kako SMARTY kreira predloške XHTML-a.
3. Filozofija prezentacijskog, poslovnog i podatkovnog sloja današnjih web aplikacija.
4. 100% objektno orijentirano programiranje.
5. Detaljan opis života (runtime) jednog programa.
6. Detaljan opis 5 klasa.
7. Detaljan opis rada 5 objekata.
8. Sažetak rada objekata.

Pripreme za vježbe

Sve se vježbe nalaze u download kompletu vježbi. Možete učiti na gotovim primjerima, ili sve tipkati od nule.

Alias

Napravite isti fizički put mapa, kao što sam i ja, pa nećete morati mijenjati fizičke adrese mapa.

1. Otvorite httpd.conf (kod mene se nalazi ovdje: C:\Program Files\Apache Group\Apache2\conf)
Kako ćete naći konfiguracijsku datoteku Apachea na svom računalu?
Ovako: Start/Control Panel/Administrative Tools/Services/
Pronađite servis Apache (kod mene je to Apache2), desni klik na servis, odaberite Properties, u polju Path to executable je ovaj put u mom slučaju: "C:\Program Files\Apache Group\Apache2\bin\Apache.exe" -k runservice
Konfiguracijska datoteka httpd.conf se nalazi u: C:\Program Files\Apache Group\Apache2\conf

2. U sekciju Aliases tipkajte istaknuti tekst:

```
# Aliases: Add here as many aliases as you need (with no
limit). The format is
# Alias fakename realname
```

```
Alias /a/ "c:/a/"
Alias /a/ "c:/a"
```

```
Alias /1_knjiga/ "c:/1_knjiga/"
```

```
Alias /1_knjiga/ "c:/1_knjiga"
```

3. Pohranite promjene u httpd.conf.
4. Restartajte Apache. (Start/Control Panel/Administrative Tools/Services/
Pronađite servis Apache (kod mene je to Apache2), označite Apache
(Apache2), gore desno odaberite Restart).

Mape za vježbe

1. Kreirajte mapu 1_knjiga: C:\1_knjiga.
2. Kreirajte mapu Poglavlje_knjigagostiju:
C:\1_knjiga\Poglavlje_knjigagostiju
3. C:\1_knjiga\Poglavlje_knjigagostiju

Vježba 1

1. Kreirajte mapu k1: C:\1_knjiga\Poglavlje_knjigagostiju\k1
2. U mapi k1, kreirajte ove mape: biblioteke, cache, configs, templates,
templates_c, include.
Ovdje je novost mapa INCLUDE.
3. U mapi INCLUDE kreirajte ove dokumente u Notepadu:

konfiguracija.inc.php

```
<?php
define("SITE_ROOT", dirname(dirname(__FILE__)));
define("SMARTY_DIR", SITE_ROOT."/biblioteke/smarty/");
define("TEMPLATE_DIR", SITE_ROOT."/templates/");
define("COMPILE_DIR", SITE_ROOT."/templates_c/");
define("CONFIG_DIR", SITE_ROOT."/configs/");
?>
```

uspostavljanje_smarty.php

```
<?php
require_once SMARTY_DIR . 'Smarty.class.php';
class HomePage extends Smarty
{
function __construct()
{
$this->Smarty();
$this->template_dir = TEMPLATE_DIR;
$this->compile_dir = COMPILE_DIR;
$this->config_dir = CONFIG_DIR;
}
}
?>
```

vrh_aplikacije.php

```
<?php
```

```
require_once 'konfiguracija.inc.php';
require_once 'uspostavljanje_smarty.php';
?>
```

U mapi CONFIGS kreirajte ovaj dokument

web_site.conf

```
naslov = "Knjiga gostiju"
```

U mapu BIBLIOTEKE ide mapa Smarty

1. U mapu biblioteke kopirajte mapu Smarty iz ranijih vježbi, iz tamošnje istoimene mape 'biblioteke'.

U mapi TEMPLATES kreirajte ove dokumente

index.tpl

```
{* smarty *}
{config_load file="web_site.conf"}
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>{#naslov#}</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../../../stil.css" type="text/css" />
</head>
<body>
{include file="zaglavlje.tpl"}
</body>
</html>
```

zaglavlje.tpl

```
<h1>Knjiga gostiju</h1>
```

U mapi k1 kreirajte

1. U mapi 'k1', korijen mapi naše aplikacije, kreirajte index.php.

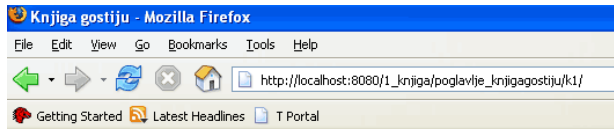
```
<?php
require_once 'include/vrh_aplikacije.php';
$home = new HomePage();
$home->display('index.tpl');

?>
```

Web stranica će prikazati ovo

1. U web pregledniku otvorite ovaj url:

http://localhost:8080/1_knjiga/poglavlje_knjigagostiju/k1/



Knjiga gostiju

templates_c

Nakon prvog učitavanja ove aplikacije, Smarty je kreirao predloške i pohranio ih u mapu `templates_c`. Slovo 'c' je ovdje od riječi 'compile'. U mapi `templates` su ne-kompajlirani predlošci i nastavak imena datoteka je `.TPL` (od riječi `template`). U mapi su kompajlirani predlošci i imaju nastavak `.php`. Riječ kompajlirani je ovdje uzeta samo kao naglasak na razlici između predložaka `.tpl` i predložaka spremnih za učitavanje `.php`. Proces kompajliranja u programskim jezicima znači kreiranje binarne datoteke, na primer `.dll`, `.exe` i `.sl`. Ovdje to nije slučaj. Datoteke `.tpl` i `.php` su tekstualne datoteke. Smarty treba od `.tpl` napraviti `.php`. To čini samo prilikom prvog učitavanja. Sljedeća učitavanja su brža jer su `.php` datoteke napravljene i čuvaju se u `templates_c`. Bilo koja promjena u kôdu, aktivirat će Smarty i napraviti će nove datoteke u `templates_c`.

Objašnjenje

U odnosu na ranije vježbe, ovdje je novost mapa `INCLUDE`. U njoj smo pohranili ove dokumente:

konfiguracija.inc.php

Pomoću php funkcije `define()`, u ovoj datoteci su definirane KONSTANTE potrebne za rad cijele aplikacije.

```
<?php
```

Najprije je definirana fizička adresa `KORIJEN` mape aplikacije. Adresa je `RELATIVNA`, pa cijelu aplikaciju možete pohranjivati na bilo koju fizičku adresu na serveru, jer su sve ostale adrese u aplikaciji definirane također `RELATIVNO` u odnosu na `korijen` mapu. To daje `MOBILNOST` našim aplikacijama.

Pogledajte u kojoj smo sad datoteci: `konfiguracija.inc.php`. Ta datoteka se nalazi u mapi `INCLUDE`. Mapa `INCLUDE` se nalazi u mapi `k1` – korijen mapi. I tu pomaže fenomenalna php funkcija `dirname()` i argument `(_FILE_)`, istaknuto u definiciji. `dirname(_FILE_)` znači `RELATIVNA` adresa tekuće stranice. To je tekuća MAPA, u našem slučaju `INCLUDE`. Argument `(_FILE_)` znači 'tekuća datoteka' u našem slučaju je to: `konfiguracija.inc.php`. Prema tome vrijedi sljedeće:

```
dirname(_FILE_) = INCLUDE
```

```
dirname(dirname(_FILE_)) = k1 (korijen mapa).
```

U funkciji `define()`, konstanti `SITE_ROOT` smo dali vrijednost `dirname(dirname(_FILE_))`:

```
define("SITE_ROOT", dirname(dirname(_FILE_)));
```

Sve ostale mape potrebne za rad Smartya, imaju ove relativne fizičke adrese u odnosu na konstantu SITE_ROOT:

```
define("SMARTY_DIR", SITE_ROOT."/biblioteke/smarty/");
define("TEMPLATE_DIR", SITE_ROOT."/templates/");
define("COMPILE_DIR", SITE_ROOT."/templates_c/");
define("CONFIG_DIR", SITE_ROOT."/configs/");
?>
```

Izraz:

SITE_ROOT."/biblioteke/smarty/

je konkatencija ili pridodavanje stringova jednih drugima. Između se stavlja točka . umjesto znaka +.

uspostavljanje_smarty.php

Smarty nam poklanja klasu Smarty. Mi najprije učitavamo datoteku na kojoj je ta klasa Smarty:

```
<?php
require_once SMARTY_DIR . 'Smarty.class.php';
```

U ovoj točki programa u memoriji imamo klasu Smarty, koja će napraviti potrebne predloške i koja nam omogućava razdvajanje php koda i XHTML koda. Klasu Smarty imamo, pa možemo napraviti našu klasu koja je naša jedina web stranica na kojoj se sve prikazuje. Zato sam tu klasu nazvao HomePage.

```
class HomePage extends Smarty
{
```

Klasa HomePage je 'extends' klasa od klase Smarty. To znači da će naslijediti sva svojstva i metode od klase Smarty. Svaka klasa nasljednica (extends) može imati svoje članove i metode. Našoj klasi HomePage dodajemo metodu __CONSTRUCT koja će se izvesti automatski čim INSTANCIRAMO objekt.

```
function __construct()
{
```

Neka se izvrši funkcija Smarty(), to je funkcija koju smo naslijedili:

```
$this->Smarty();
```

Kreiramo konstantu template_dir i pohranjujemo je u RAM servera, jer ta konstanta je fizički put do mape TEMPLATE. Taj put je potreban za rad našeg objekta koji mora 'kompajlirati' sve .tpl datoteke koje pronade u mapi TEMPLATE:

```
$this->template_dir = TEMPLATE_DIR;
```

Kreiramo konstantu compile_dir i pohranjujemo je u RAM poslužitelja:

```
$this->compile_dir = COMPILE_DIR;
```

Kreiramo konstantu config_dir i pohranjujemo je u RAM poslužitelja:

```
$this->config_dir = CONFIG_DIR;
}
}
```

```
?>
```

I to je sve u ovoj najvažnijoj klasi za našu aplikaciju.

vrh_aplikacije.php

```
<?php
```

Ovu datoteku smo kreirali za vrh naše index.php stranice. U ovoj datoteci ćemo pozivati sve klase koje se moraju učitati u RAM prije svih drugih. U ovoj vježbi nam trebaju samo ovi dokumenti. Na prvom su definirane KONSTANTE potrebne za rad naše aplikacije:

```
require_once 'konfiguracija.inc.php';
```

a na drugom je klasa Smarty i naša extends klasa HomePage:

```
require_once 'uspostavljanje_smarty.php';  
?>
```

Mapa k1

index.php

```
<?php
```

Učitavamo u RAM poslužitelja datoteku vrh_aplikacije.php. Time smo učitali sve potrebne klase.

```
require_once 'include/vrh_aplikacije.php';
```

INSTANCIRAMO objekt HomePage (new HomePage) i dodjeljujemo ga varijabli \$home. Pohranjujemo ga u RAM u varijabli \$home:

```
$home = new HomePage();
```

Pozivamo metodu display() neka kreira web stranicu na bazi predloška index.tpl:

```
$home->display('index.tpl');
```

```
?>
```

Mapa templates

U mapi templates su samo dvije datoteke: index.tpl i zaglavlje.tpl.

index.tpl

Ima samo ovaj zanimljiv kôd. U ovom retku je prikazano kako se ugrađuju Smarty varijable. Komentar u Smartyu se piše unutar vitičastih zagrada i zvjezdice {*...*}. Na vrhu smo naveli u komentaru da gledamo u smarty predložak. To je dobra praksa, pomaže puno programeru u brzem snalaženju kod velikih aplikacija.

```
{* smarty *}
```

Sljedi Smarty sintaksa za učitavanje datoteke web_site.conf u kojoj se za sad nalazi samo varijabla 'naslov':

```
{config_load file="web_site.conf"}
```

Ugrađujemo vrijednost varijable #naslov. Vrijednost smo odredili u datoteci web_site.conf, koju smo upravo učitali u RAM.

```
<title>{#naslov#}</title>
```

Učitavamo datoteku "zaglavlje.tpl".

```
{include file="zaglavlje.tpl"}
```

Zaključak

I u ovoj vježbi, kao i u ranijim vježbama Smarty, imamo dva glavna igrača: index.tpl i index.php. Prvi je čisti XHTML, drugi je čisti PHP. Ovdje još nemamo varijabli na relaciji index.tpl i index.php, pa nam nije trebala metoda ASSIGN() koja povezuje varijable Smartya i naš sadržaj. U ovoj vježbi mi i nemamo sadržaja.

U ovoj vježbi smo se više zabavili stvaranju UČINKOVITOG i DJELOTVORNOG programerskog OKRUŽENJA. Što to znači? To znači da smo našu aplikaciju podijelili na više datoteka, ali s nekom logikom. Konkretno rečeno, index.tpl smo razdvojili u više .tpl datoteka i index.php smo razdvojili u više .php datoteka. Ta logika je logika bolje organiziranosti, pa ćemo se u svakom trenutku brže snalaziti. Na primjer, ako trebamo dodati novu konstantu ili promijeniti postojeću, otvorit ćemo datoteku konfiguracija.inc.php i učiniti to u njoj.

Ako treba predučitati neku novu klasu, dodat ćemo instrukciju u datoteku vrh_aplikacije.php. Ako želimo dodati neke varijable koje se ne mijenjaju dinamički na našoj Home Page, učinit ćemo to u datoteci web_site.conf. Ako naš centralni objekt HomePage u index.php treba imati dodatne metode i članove, mi ćemo ih dodati u extends klasu HomePage u datoteci uspostavljanje_smarty.php.

Toliko smo se dobro organizirali da nam **index.php** sadrži SAMO ovaj kôd:

```
<?php
require_once 'include/vrh_aplikacije.php';
$home = new HomePage();
$home->display('index.tpl');

?>
```

Neće se puno povećati količina kôda niti u sljedećoj vježbi, kad ćemo posegnuti po sadržaj iz baze podataka. Kako naša aplikacija raste, tako ćemo sve više upoznavati blagodati Smartya i objektno orijentiranog programiranja i ove naše logičke organiziranosti!

Idemo po sadržaj iz MySQL baze.

Vježba 2

Priprema za vježbu 2

PEAR

PEAR je PHP tehnologija kao Smarty, samo na drugom kraju web aplikacije i s drugom namjenom. PEAR je dodatak PHP-u za učinkovitije, djelotvornije i

mobilnije spajanje php aplikacija na baze podataka. Učinkovitije zato što ćemo morati samo na jednom mjestu promijeniti konstante potrebne za spoj na bazu. Djelotvornije zato što ćemo moći mijenjati tehnologije baza podataka (MySQL, SQL Server, Access itd), bez da interveniramo u programski kôd u onoj mjeri u kojoj bi morali bez PEAR-a. Mogućnost da našu web aplikaciju možemo spojiti na bilo koju tehnologiju poslužitelja baza podataka bez dodatnih sati rada, zovemo mobilnošću naše aplikacije. Sve tehnologije baza podataka koriste SQL skriptni jezik. Ali svaka ima svoj dijalekt. Kad koristimo PEAR, interвенicije radi dijalekta su minimalne ili nikakve.

Instalacija PEAR-a

1. PEAR je otvoreni kôd, dakle besplatan i možete sve o njemu naučiti i skinuti ga na ovoj adresi:

<http://pear.php.net/>

Za rad uz ovaj obrazovni sadržaj preporučujem kopiranje PEAR-a kojeg sam ja koristio, pa tako neće imati nikakve probleme oko nesuglasnosti verzija.

2. Zamijenite mapu 'biblioteke' u k1 s istoimenom mapom iz k2. U novoj mapi 'biblioteke' nalazi se i Smarty i PEAR.

I to je to. Nikakve dodatne radnje nisu potrebne. Smarty i PEAR su samo paketi PHP klasa i PHP skripti koje je dovoljno kopirati tamo gdje vam trebaju.

Kreiranje baze podataka 'knjigagostiju'

1. Napišite ovaj dokument u Notepadu:

```
DROP DATABASE IF EXISTS knjigagostiju;

CREATE DATABASE knjigagostiju;

USE knjigagostiju;

DROP TABLE IF EXISTS knjigagostiju;

CREATE TABLE knjigagostiju (
    id int(11) NOT NULL auto_increment,
    Ime varchar(255) NOT NULL default '',
    Datum datetime NOT NULL default '0000-00-00 00:00:00',
    Komentar text NOT NULL,
    PRIMARY KEY (id),
    KEY Datum (Datum)
) TYPE=MyISAM;

GRANT ALL PRIVILEGES ON knjigagostiju.* TO admin@localhost
IDENTIFIED BY 'admin' WITH GRANT OPTION;

INSERT INTO knjigagostiju VALUES (1, 'Vinko', '2006-08-14',
'Sviđa mi se raditi po knjizi.);
```



```
INSERT INTO knjigagostiju VALUES (2, 'Vinko', '2006-08-15',  
'Smarty je zakon.');
```

```
INSERT INTO knjigagostiju VALUES (3, 'Vinko', '2006-08-16',  
'PEAR je odličan.');
```

```
INSERT INTO knjigagostiju VALUES (4, 'Vinko', '2006-09-16',  
'Sa Smartyem i PEAR imamo puno manje kodiranja');
```

2. Pohranite ga pod imenom knjigagostiju.sql u mapu bin u kojoj se nalazi izvršna datoteka MySQL naredbodavnog retka. U mom slučaju je put do te datoteke ovaj:

```
C:\Program Files\MySQL\MySQL Server 4.1\bin
```

3. Otvorite Command Prompt, dođite do mape bin, što u mom slučaju izgleda ovako:

```
C:\Program Files\MySQL\MySQL Server 4.1\bin
```

4. U naredbodavnom retku ...bin> tipkajte ovu naredbu:

```
mysql -u root < knjigagostiju.sql
```

5. ili u mom slučaju

```
mysql -u root@localhost < knjigagostiju.sql
```

Sad imamo kreiranu bazu podataka 'knjigagostiju', s nekoliko redaka unosa. Možemo se posvetiti pisanju naše PHP aplikacije 'Knjiga gostiju'.

Kraj priprema za ovu vježbu.

Što će nam prikazati web stranica na kraju ove vježbe

Web stranica index.php, na kraju ove vježbe će nam prikazati tekući datum i sve retke iz baze 'knjigagostiju':



Projektiranje web aplikacije u tri sloja

Današnje profesionalne web aplikacije su rađene u tri sloja: prezentacijski, poslovni i podatkovni. To ćemo i mi napraviti. Sva tri sloja će biti 100% objektno orijentirana. Uz to, prezentacijski sloj će imati razdvojen PHP od XHTML-a pomoću Smartya, a podatkovni sloj će imati PEAR za spoj na bazu.

Podatkovni sloj

Početi ćemo od podatkovnog sloja, jer nam taj sloj donosi sadržaj iz baze. Evo stranice koju trebate kreirati od nule za podatkovni sloj.

1. Kreirajte mapu podatkovni_objekti u mapi k1.
2. U mapi podatkovni_objekti kreirajte od nule u Notepadu datoteku podatkovni_objekt.php:

podatkovni_objekt.php

```
<?php
class PodatkovniObjekt
{
    function __construct()
    {
        $this->bPodataka = $GLOBALS['gBazaPodataka'];
    }
    public function GetKnjigaGostiju()
    {
        $query_string = "SELECT id, Ime, Datum, Komentar FROM
knjigagostiju";
        $result = $this->bPodataka->BpGetAll($query_string);
        return $result;
    }
}
?>
```

Poslovni sloj

Poslovni sloj je super izolirani sloj. Od posjetitelja ga štiti prezentacijski sloj, a od baze podataka ga štiti podatkovni sloj. Čemu takva zaštita, izolacija tog sloja i čemu uopće služi? U našim primjerima će imati malo posla, jer mi u aplikaciji Knjiga gostiju koju upravo radimo nećemo imati neku poslovnu politiku ili poslovna pravila. A upravo za to je namijenjen srednji sloj aplikacije, pa mu od tud i ime dolazi: poslovni sloj.

1. Kreirajte mapu poslovni_objekti u mapi k1.
2. Kreirajte od nule datoteku poslovni_objekt.php:

poslovni_objekt.php

```
<?php
require_once SITE_ROOT .
'/podatkovni_objekti/podatkovni_objekt.php';
class PoslovniObjekt
{
    private $mPodatkovniObjekt;
    function __construct()
    {
        $this->mPodatkovniObjekt = new PodatkovniObjekt();
    }
}
```

```

public function GetKnjigaGostiju()
{
    $result = $this->mPodatkovniObjekt->GetKnjigaGostiju();
    return $result;
}
?>

```

Prezentacijski sloj

Prezentacijski sloj čine datoteke u mapi templates. Do sad su nam bile dovoljne .tpl datoteke. Od sad nam trebaju i posebne .php datoteke uz neke .tpl datoteke.

knjiga_gostiju.tpl

Ovu datoteku napravite od nule u mapi templates.

```

{* knjiga_gostiju.tpl *}
{load_knjiga_gostiju assign="knjiga_gostiju"}
{section name=i loop=$knjiga_gostiju->mKnjigaGostiju}
{$smarty.now|date_format:"%d.%m.%Y."}
    {$knjiga_gostiju->mKnjigaGostiju[i].id}
    {$knjiga_gostiju->mKnjigaGostiju[i].Ime}
    {$knjiga_gostiju->mKnjigaGostiju[i].Datum|date_format:"%d.%m.%Y."}
    {$knjiga_gostiju->mKnjigaGostiju[i].Komentar}
{/section}

```

function.load_knjiga_gostiju.php

Ovo je novost za nas u učenju Smartya. Uz gornju datoteku knjiga_gostiju.tpl potrebno je kreirati i datoteku function.load_knjiga_gostiju.php. Ovoj datoteci moramo dati ime koje počinje s **function.load_**. Slijedi ime datoteke knjiga_gostiju pa nastavak .php: function.load_knjiga_gostiju.php. Ova pravila diktira Smarty. A to će nam razdvojiti PHP od XHTML.

Zašto su potrebne dvije datoteke, a ranije nam je bila potrebna samo jedna i to s nastavkom .tpl? Zato, što smo prije u jednostavnim primjerima, funkciju ASSIGN() pisali u kôdu index.php. Sad je u index.php ostala samo metoda DISPLAY('index.tpl'), a metoda ASSIGN() je dobila svoju posebnu PHP datoteku čije ime počinje s **function.load_**. To je zato što sad treba više kôda za kreiranje Smarty varijabli i dodjeljivanja sadržaja tim varijablama.

1. Napravite u k1 posebnu mapu smarty_plugins za sve datoteke tipa function.load.
2. U mapi smarty_plugins napravite novu datoteku i dajte joj ime function.load_knjiga_gostiju.php:

```

<?php
function smarty_function_load_knjiga_gostiju($params,
$smarty)
{
    $knjiga_gostiju = new KnjigaGostiju();
    $knjiga_gostiju->init();
}

```

```

    $smarty->assign($params['assign'], $knjiga_gostiju);
}
class KnjigaGostiju
{
    public $mKnjigaGostiju;
    private $mPoslovniObjekt;

    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();
    }
    function init()
    {
        $this->mKnjigaGostiju = $this->mPoslovniObjekt-
        >GetKnjigaGostiju();
    }
}
?>

```

index.tpl

1. Otvorite postojeću datoteku index.tpl i dodajte naglašeni kôd:

```

{* smarty *}
{config_load file="web_site.conf"}
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>{#naslov#}</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../stil.css" type="text/css" />
</head>
<body>
{include file="zaglavlje.tpl"}
{include file="knjiga_gostiju.tpl"}

</body>
</html>

```

zaglavlje.tpl

Ostaje nepromijenjeno.

Ostale potrebne nove datoteke i promjene na postojećim datotekama

index.php

```

<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';

```

```
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>
```

vrh_aplikacije.php

```
<?php
require_once 'konfiguracija.inc.php';
require_once 'ustpostavljanje_smarty.php';
require_once 'bazapodataka.php';

//Instanciranje globalnog objekta BazaPodataka
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
?>
```

ustpostavljanje_smarty.php

```
<?php
require_once SMARTY_DIR . 'Smarty.class.php';
class HomePage extends Smarty
{
function __construct()
{
$this->Smarty();
$this->template_dir = TEMPLATE_DIR;
$this->compile_dir = COMPILE_DIR;
$this->config_dir = CONFIG_DIR;

$this->plugins_dir[0] = SMARTY_DIR . 'plugins';
$this->plugins_dir[1] = SITE_ROOT . "/smarty_plugins";
}
}
?>
```

konfiguracija.inc.php

```
<?php
//Poglavlje prvo o knjizi gostiju
define("SITE_ROOT", dirname(dirname(_FILE_)));
define("SMARTY_DIR", SITE_ROOT."/biblioteke/smarty/");
define("TEMPLATE_DIR", SITE_ROOT."/templates/");
define("COMPILE_DIR", SITE_ROOT."/templates_c/");
define("CONFIG_DIR", SITE_ROOT."/configs/");

//Poglavlje drugo o knjizi gostiju
//Ovaj kod će otkriti o kojem se operacijskom sustavu radi
//i primjeniti odgovarajuće znakove u include_path.
//Za to koristi ini_set PHP funkciju
if ((substr(strtoupper(PHP_OS), 0, 3)) == "WIN")
    define("PATH_SEPARATOR", "\\");
else
    define("PATH_SEPARATOR", ":");
ini_set('include_path', SITE_ROOT . '/biblioteke/PEAR' .
PATH_SEPARATOR . ini_get('include_path'));
//podaci za logiranje u bazu
```

```

define("USE_PERSISTENT_CONNECTIONS", "true");
define("DB_SERVER", "localhost");
define("DB_USERNAME", "admin");
define("DB_PASSWORD", "admin");
define("DB_DATABASE", "knjigagostiju");
define("MYSQL_CONNECTION_STRING", "mysql://" . DB_USERNAME .
":" .
    DB_PASSWORD . "@" . DB_SERVER . "/" . DB_DATABASE);
?>

```

dno_aplikacije.php

Dodajte ovu novu datoteku u mapu include.

```

<?php
$GLOBALS['gBazaPodataka']->BpDisconnect();
?>

```

bazapodataka.php

Dodajte ovu novu datoteku u mapu include.

```

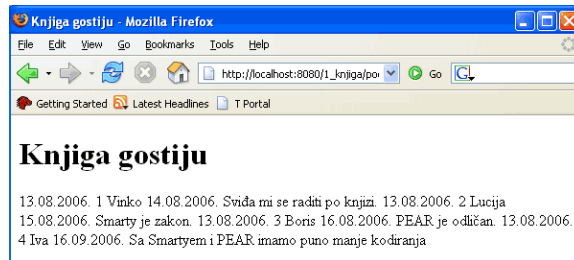
<?php
require_once 'DB.php';
class BazaPodataka
{
    public $bp;
    function __construct($connectionString)
    {
        $this->bp = DB::connect($connectionString,
                                USE_PERSISTENT_CONNECTIONS);
        $this->bp->setFetchMode(DB_FETCHMODE_ASSOC);
    }
    public function BpDisconnect()
    {
        $this->bp->disconnect();
    }
    public function BpQuery($queryString)
    {
        $result = $this->bp->query($queryString);
        return $result;
    }
    public function BpGetAll($queryString)
    {
        $result = $this->bp->getAll($queryString);
        return $result;
    }
    public function BpGetRow($queryString)
    {
        $result = $this->bp->getRow($queryString);
        return $result;
    }
    public function BpGetOne($queryString)
    {
        $result = $this->bp->getOne($queryString);
        return $result;
    }
}

```

```
}
}
?>
```

To je sve u ovoj vježbi. Pogledajte rezultate vašeg truda.

Otvorite u web pregledniku aplikaciju Knjiga gostiju



Odlično. Na stranici je sadržaj iz baze 'knjigagostiju'! Još samo objašnjenje kako je taj sadržaj stigao do stranice!

Objašnjenje

Objasnit ćemo programski kôd instrukciju po instrukciju, onim redom kojim su te instrukcije pristizale u RAM pa u PROCESOR na izvršenje. Kad procesor izvrši (procesuira) instrukciju, rezultat pohranjuje u RAM i uzima iz RAM-a sljedeću instrukciju i izvršava je.

Pa krenimo redom od trenutka kad smo upisali url u adresno polje web preglednika i udarili ENTER.

Put kojim je sadržaj stigao iz baze do web stranice

index.php

1. Pozvali smo ovaj URL naše aplikacije:

`http://localhost:8080/1_knjiga/poglavlje_knjigagostiju/k1/`

To je poziv stranici `index.php`:

```
<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>
```

2. Stranica `index.php` se izvršava ovako:

- a. poziva na izvršavanje stranicu: `include/vrh_aplikacije.php`
- b. poziva na izvršavanje stranicu: `poslovni_objekti/poslovni_objekt.php`

- c. INSTANCIRA objekt HomePage:

```
$home = new HomePage();
```

- d. poziva na izvršavanje metodu DISPLAY()

```
$home->display('index.tpl');
```

- e. poziva na izvršavanje stranicu: include/dno_aplikacije.php

U stavkama a. i b. treba se dogoditi kontakt s bazom i dovlačenje i pohranjivanje u RAM podataka iz baze. Jer, u točki c. se rađa naš objekt, koji poziva svoju metodu display() i počinje kreacija prezentacijskog sloja u kojeg Smarty ugrađuje podatke iz RAM-a.

3. Što je ostalo u RAM-u nakon procesuiranja php kôda vrh_aplikacije.php?

vrh_aplikacije.php

```
<?php
```

- a. Poziva na izvršenje konfiguracija.inc.php:

```
require_once 'konfiguracija.inc.php';
```

- b. Poziva na izvršenje uspostavljanje_smarty.php:

```
require_once 'uspostavljanje_smarty.php';
```

- c. Poziva na izvršenje bazapodataka.php.php:

```
require_once 'bazapodataka.php';
```

- d. Inicijalizira i pohranjuje u varijabli \$gBazaPodataka globalni objekt BazaPodataka:

```
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
```

```
?>
```

konfiguracija.inc.php:

- a. Definira konstante koje su potrebne SMARTY-ju za izradu predložaka prezentacijskog sloja:

```
<?php
```

```
define("SITE_ROOT", dirname(dirname(__FILE__)));
define("SMARTY_DIR", SITE_ROOT."/biblioteke/smarty/");
define("TEMPLATE_DIR", SITE_ROOT."/templates/");
define("COMPILE_DIR", SITE_ROOT."/templates_c/");
define("CONFIG_DIR", SITE_ROOT."/configs/");
```

- b. Ovisno o kojem se operacijskom sustavu radi na poslužitelju, Windows ili Linux (Unix), Mijenja konfiguracijsku postavku 'include_path' u konfiguracijskoj datoteci PHP-a, php.ini, koja se nalazi u mapi Windows. Mijenja PATH_SEPARATOR od : u ; i obratno, ovisno o OS-u. Ako je OS Windows, stavlja ;, u suprotnom stavlja :.

```
if ((substr(strtoupper(PHP_OS), 0, 3)) == "WIN")
    define("PATH_SEPARATOR", ";");
```



```
else
    define("PATH_SEPARATOR", ":");
```

- c. Odabrani PATH_SEPARATOR pomoću metode ini_set(), dodijeljuje skriptama PEAR paketa, a onda te skripte prijavljuje PHP konfiguracijskoj datoteci pomoću metode ini_get().

```
ini_set('include_path', SITE_ROOT . '/biblioteke/PEAR' .
PATH_SEPARATOR . ini_get('include_path'));
```

- d. Postavlja konstantu USE_PERSISTENT_CONNECTIONS na 'true'. To znači da će spoj na bazu podataka biti aktivan cijelo vrijeme procesuiranja stranice index.php. Kreirat će se na početku kôda, a zatvorit ćemo ga na kraju kôda. Na taj način će isti spoj baze poslužiti svim pozivima prema bazi.

```
define("USE_PERSISTENT_CONNECTIONS", "true");
```

- e. U RAM poslužitelja se pohranjuje adresa servera, u našem slučaju je to 'localhost':

```
define("DB_SERVER", "localhost");
```

- f. U RAM poslužitelja se pohranjuje vrijednost 'admin' kao korisničko ime i zaporka 'admin'. To korisničko ime i zaporku smo kreirali pomoću ovog SQL-a u dokumentu knjigagostiju.sql:
GRANT ALL PRIVILEGES ON knjigagostiju.* TO
admin@localhost IDENTIFIED BY 'admin' WITH GRANT
OPTION;

```
define("DB_USERNAME", "admin");
define("DB_PASSWORD", "admin");
```

- g. U RAM poslužitelja se pohranjuje vrijednost imena baze podataka:

```
define("DB_DATABASE", "knjigagostiju");
```

- h. U RAM poslužitelja se pohranjuje vrijednost stringa za spoj na našu bazu 'knjigagostiju'. String se sastoji od više stringova i konstanti. Svaki string je pod navodnicima, slijedi točka i opet string u navodnicima i sve tako. To je konkatencija ili zbrajanje stringova. Oko konstanti se ne stavljaju navodnici.

```
define("MYSQL_CONNECTION_STRING", "mysql://" . DB_USERNAME .
":" .
    DB_PASSWORD . "@" . DB_SERVER . "/" . DB_DATABASE);
?>
```

U ovoj točki procesuiranja naše aplikacije imamo u RAM-u pohranjene sve potrebne konstante za spoj na bazu podataka 'knjigagostiju'. Super. Spremni smo za programski kôd koji će upotrijebiti te konstante i spojiti nas na bazu.

uspostavljanje_smarty.php:

- a. Učitavamo u RAM poslužitelja klasu Smarty:

```
<?php
```

```
require_once SMARTY_DIR . 'Smarty.class.php';
```

- b. Kreiramo extends klasu HomePage i učitavamo je u RAM poslužitelja:

```
class HomePage extends Smarty
{
function __construct()
{
$this->Smarty();
$this->template_dir = TEMPLATE_DIR;
$this->compile_dir = COMPILE_DIR;
$this->config_dir = CONFIG_DIR;
```

- c. Ovo je dodani kôd. Ovaj kôd samo pohranjuje još dvije konstante u RAM poslužitelja. Prva konstanta plugins_dir[0] sadrži fizički put do mape 'plugins' u mapi Smarty:

```
$this->plugins_dir[0] = SMARTY_DIR . 'plugins';
```

- d. i konstanta plugins_dir[1] sadrži fizički put do mape smarty_plugins koju smo mi kreirali:

```
$this->plugins_dir[1] = SITE_ROOT . "/smarty_plugins";
}
}
?>
```

U ovoj točki procesuiranja naše PHP aplikacije Knjiga gostiju, još uvijek nema KONKRETNOSTI tj. nema objekata. Na razini smo APSTRAKTNOSTI tj. klasa koje učitavamo u RAM. Tako mora biti. Klase su predlošci za objekte. Klase ništa ne rade. Kad treba nešto napraviti, moramo INSTANCIRATI i INICIJALIZIRATI objekt i onda će taj objekt obaviti posao. Pozvat će svog člana, pozvat će svoju metodu i sve tako. Ako treba pozvat će drugog objekta, a ovaj opet nekog drugog i tako će objekti obaviti sve, baš sve što treba. Zato ih zovemo KONKRETNOST. Cijelo to vrijeme klase nepomično leže u ladicama RAM-a i apsolutno ništa ne rade. Zato ih zovemo APSTRAKCIJA.

bazapodataka.php.php:

- a. Opet samo učitavanje klasa u RAM poslužitelja. Najprije se učita klasa DB. To je klasa koju smo dobili u paketu PEAR:

```
<?php
require_once 'DB.php';
```

- b. A ovo je klasa koju smo mi napisali. Objekt koji se instancira na njenu sliku i priliku drži 'ključeve trezora'. Drži ključeve baze podataka. Ovom objektu će pristupati SAMO naš PODATKOVNI sloj. Pozivat će ga kad treba napraviti spoj na poslužitelj baza MySQL.

Pozivat će ga kad treba pristupiti bazi 'knjigagostiju'. Pozivat će ga za bilo koji UPIT koji iz aplikacije dođe u PODATKOVNI sloj. I još nešto važno. Kao što SAMO podatkovni sloj ima pristup ovom objektu 'menadžeru baze

podataka', tako isto SAMO poslovni sloj ima pristup podatkovnom sloju.

Mora se znati red, zar. Jer, web aplikacija je firma, organizacija koja ima svoje poslove i točno se zna tko što radi i tko s kim komunicira. Tko radi u toj firmi? Rade objekti koje smo napravili na sliku i priliku svoje civilizacije.

```
class BazaPodataka
{
```

- c. Klasi kreiramo javnog člana ili javno svojstvo \$bp:

```
public $bp;
```

- d. Klasa će imati i metodu CONSTRUCT. To je ona metoda koja se IZVRŠI automatski čim se INSTANCIRA objekt. Metoda CONSTRUCT ima jedan jedini argument, to je varijabla \$connectionString. To znači da će ova metoda obaviti spoj na MySQL i spoj na našu bazu 'knjigagostiju'. Preko ovog argumenta će se prenijeti konstanta MYSQL_CONNECTION_STRING, kao vrijednost varijabli unutar metode construct.

```
function __construct($connectionString)
{
```

- e. Pseudonim klase \$this poziva svog člana bp na lijevoj strani znaka jednakosti. Sintaksa nalaže da ime člana bp pišemo ovdje BEZ znaka dolara (\$) ispred imena. U ne objektno orijentiranom programiranju bismo jednostavno pisali \$bp. Ovdje je to malo drukčije, ali kad se naučite bit će vam jako normalno.

Na desnoj strani znaka jednakosti koristimo operator dvostruka dvotočka :: za dohvat KONSTANTE u nekoj vanjskoj klasi. Ta vanjska klasa se zove DB, već je učitana u RAM i dobili smo je u paketu RAM. Dakle, klasa DB:: poziva svoju metodu CONNECT(). Metoda connect ima dva argumenta ili parametra. Prvi parametar je varijabla \$connectionString preko koje će izvana stići string za spoj na bazu, a drugi parametar je konstanta USE_PERSISTENT_CONNECTIONS koja je već u RAM-u i kojoj smo dali vrijednost 'true'.

Zaključak: ova instrukcija nas spaja na bazu podataka i drži tu vezu s bazom sve do posljednje instrukcije na stranici index.php. Još točnije rečeno, u RAM pohranjujemo člana bp kojem smo dodijelili vrijednost resursa spoja na bazu podataka. Spoj na bazu podataka je PHP tip podataka resurs.

```
$this->bp = DB::connect($connectionString,
                        USE_PERSISTENT_CONNECTIONS);
```

- f. Član bp poziva i pohranjuje metodu `setFetchMode()` koja od podataka iz baze pravi ARRAY. Kao parametar je navedena konstanta `DB_FETCHMODE_ASSOC` što znači, ne pravi obični array nego ASOCIJATIVNI ARRAY o kojem smo detaljno učili ranije u ovoj knjizi.

```
$this->bp->setFetchMode(DB_FETCHMODE_ASSOC);
}
```

Kraj konstruktora. Malo kôda, ali će puno napraviti u svom instanciranom objektu.

- g. U našoj klasi BazaPodataka, koju smo mogli nazvati i klasa 'UpraviteljBazePodataka', s obzirom na funkciju koja joj je namjenjena, sad gradimo sve METODE koje će joj trebati za upravljanje s bazom podataka. Konstruktor je automatski otvorio vrata baze podataka, a tko će zatvoriti ta vrata? Napravimo metodu koja će uništiti resurs 'spoj na bazu', točnije, koja će u RAM-u uništiti člana \$bp, kojeg je stvorio konstruktor. Neka se metoda zove `BpDisconnect()` i mora biti javna da je mogu pozivati i objekti izvan nje:

```
public function BpDisconnect()
{
```

- h. Vrlo jednostavno je kad imamo gotovu php funkciju koja razara spoj na bazu. To je funkcija `disconnect()`. Ovo je dobro. Član bp poziva metodu `disconnect()` koja prekida vezu s bazom.

```
$this->bp->disconnect();
}
```

- i. Sad ćemo našoj klasi BazaPodataka kreirati metode za sve moguće tipove UPITA. Tipovi upita su definirani metodama u PEAR klasi DB. Počnimo od metode `query()` koju ćemo koristiti za upite INSERT, DELETE i UPDATE.

Još nešto. Naša klasa BazaPodataka, u kojoj se još nalazimo, pristupat će metodama PEAR-a preko svojih metoda.

Napravit će svoju metodu, npr. `BpQuery()`, koja će u sebi sadržavati poziv PEAR-ove metode `query()`. Takve omotače ćemo kreirati za sve metode koje slijede:

```
public function BpQuery($queryString)
{
```

```
    $result = $this->bp->query($queryString);
```

- j. 'return' nam vraća podatke koji zadovoljavaju upit. Svaka od ovih funkcija će imati i taj redak s return.

```
    return $result;
}
```

- k. PEAR metoda `getAll()` će nam vratiti sve retke koji zadovoljavaju upit. Umotali smo je u našu metodu kojoj smo dali ime `BpGetAll()`:

```
public function BpGetAll($queryString)
{
    $result = $this->bp->getAll($queryString);
    return $result;
}
```

- l. PEAR metoda `getRow()` će nam vratiti samo prvi redak od svih mogućih koji zadovoljavaju upit. Umotali smo je u našu metodu kojoj smo dali ime `BpGetRow()`:

```
public function BpGetRow($queryString)
{
    $result = $this->bp->getRow($queryString);
    return $result;
}
```

- m. PEAR metoda `getOne()` će nam vratiti samo prvi stupac prvog retka od svih mogućih koji zadovoljavaju upit. Umotali smo je u našu metodu kojoj smo dali ime `BpGetOne()`:

```
public function BpGetOne($queryString)
{
    $result = $this->bp->getOne($queryString);
    return $result;
}

?>
```

Inicijaliziranje prvog objekta

Konačno smo došli i do prve KONKRETNOSTI u našoj web aplikaciji. Dolazi do inicijalizacije objekta `BazaPodataka`:

```
new BazaPodataka(MYSQL_CONNECTION_STRING)
```

Ako podignete pogled vidjet ćete da smo upravo analizirali klasu `BazaPodataka` i da klasa ima konstruktor koji će odmah prilikom INSTANCIRANJA i INICIJALIZIRATI objekt. Inicijalizirati objekt znači dodijeliti mu početnu vrijednost. Konstruktor ima jedan parametar preko kojeg prenosimo vrijednosti IZVANA u konstruktor. Tu vrijednost pišemo u parametar klase:

`new BazaPodataka(MYSQL_CONNECTION_STRING)`. U našem slučaju ta vrijednost je konstanta `MYSQL_CONNECTION_STRING`. Tako smo uhvatili moment kad se kreira prvi objekt i kad se tom objektu dodjeljuje vrijednost izvana. Ta vrijednost se automatski dodjeljuje varijabli konstruktora `$connectionString`. Konstruktor pomoću te vrijednosti ostvaruje spoj s bazom 'knjigagostiju' i pohranjuje taj spoj u RAM u varijabli `$db`.

```
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
```

To je instrukcija u kojoj se instancirao naš prvi objekt i to objekt koji je ostvario automatski spoj s bazom. Inicijalizacija ili početna vrijednost tog objekta je pohranjena u RAM u varijabli \$gBazaPodataka. Sve sljedeće instrukcije je mogu tamo dohvatiti.

Podsjetimo se do kud smo stigli u procesuiranju naše aplikacije

Stigli smo do mjesta u aplikaciji koje sam označio bold tekstom. U memoriju poslužitelja je učitani php dokument poslovni_objekt.php.

index.php

```
<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>
```

poslovni_objekt.php

- a. Prije svega, poziva se podatkovni_objekt.php i tako se u memoriju učitava klasa PodatkovniObjekt.

```
<?php
require_once SITE_ROOT .
'/podatkovni_objekti/podatkovni_objekt.php';
```

podatkovni_objekt.php

- a. U memoriju se učitava klasa PodatkovniObjekt

```
<?php
class PodatkovniObjekt
{
    function __construct()
```

- b. Automatski u metodi construct() se globalna varijabla gBazaPodataka, koja u sebi sadrži prvi objekt nastao u našoj aplikaciji, dodjeljuje varijabli bPodataka:

```
{
    $this->bPodataka = $GLOBALS['gBazaPodataka'];
}
```

- c. U klasi kreiramo metodu GetKnjigaGostiju() koja će nam realizirati željeni upit:

```
public function GetKnjigaGostiju()
{
```

- d. Naš SQL upit SELECT dodjeljujemo varijabli na lijevoj strani znaka jednakosti:

```
    $query_string = "SELECT id, Ime, Datum, Komentar FROM
knjigagostiju";
```

- e. Varijabli na lijevoj strani jednakosti dodjeljujemo podatke tipa resurs. Ono što vrati naš upit iz baze je tip podataka resurs. Na desnoj strani bi u VB.NET jeziku pisalo `bPodataka.BpGetAll($query_string)`. Ovdje umjesto točke (dot) imamo strelicu `->` i pseudo objekt tekuće klase `$this`.

Član `bPodataka` u sebi je pohranio vezu s bazom. Taj član ima pristup svim metodama objekta `BazaPodataka`, jer smo gore već tom članu dodijelili cijelu globalnu varijablu `gBazaPodataka` koja mu je donijela i sve metode objekta `BazaPodataka`.

Zaključak: `bPodataka` je objekt klase `BazaPodataka`. Taj objekt poziva svoju metodu `BpGetAll`:

```
$result = $this->bPodataka->BpGetAll($query_string);
```

koja mu vraća sve retke iz baze.

```
return $result;
}
}
?>
```

Upravo se samo UČITALA klasa `PodatkovniObjekt` i ja sam opisao što se događa kad se od te klase INSTANCIRA objekt.

Međutim, taj objekt se još nije dogodio i klasa `PodatkovniObjekt` sadrži apstraktne podatke u memoriji. Klasa nije konkretizirana tj. nema još svog objekta.

Sad je na redu izvršenje nastavka kôda dokumenta `poslovni_objekt.php`.

poslovni_objekt.php NASTAVAK

- f. Kreiramo klasu `PoslovniObjekt`

```
class PoslovniObjekt
{
```

- g. koja će imati jednog privatnog člana

```
private $mPodatkovniObjekt;
```

- h. i konstruktor:

```
function __construct()
{
```

- i. Svom jedinom članu dodjeljuje instancirani objekt klase `PodatkovniObjekt`.

```
$this->mPodatkovniObjekt = new PodatkovniObjekt();
```

```
}
```

- j. Klasi smo kreirali i metodu koja samo poziva istoimenu metodu u podatkovnom sloju:

```
public function GetKnjigaGostiju()
{
    $result = $this->mPodatkovniObjekt->GetKnjigaGostiju();
```

```

        return $result;
    }
}
?>

```

Na ovom mjestu procesuiranja naše aplikacije

mi još uvijek imamo u MEMORIJI poslužitelja SAMO jedan jedini objekt i to je globalni objekt gBazaPodataka. Pošto je globalan, dohvatljiv je iz svih pozicija naše aplikacije.

Ali smo zato napunili RAM poslužitelja sa svim potrebnim klasama. Posljednje učitane klase su PodatkovniObjekt i PoslovniObjekt.

Ulazimo u finalnu fazu procesuiranja naše aplikacije. Pogledajmo gdje smo stigli pa nastavimo s objašnjenjem.

Gdje smo stigli?

Označio sam bold tekстом mjesto do kojeg smo stigli u našoj aplikaciji.

index.php

```

<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>

```

Nastavak objašnjenja

index.php

```

<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';

```

- a. Evo drugog objekta koji se INSTANCIRAO u našoj aplikaciji. To je objekt koji je personifikacija naše HomePage, pa sam mu i po tome dao ime 'home':

```
$home = new HomePage();
```

- b. Naš objekt nasljeđuje sve od klase Smarty. Tako on poziva svoju metodu display('index.tpl') s argumentom index.tpl. Time je pokrenut proces izgradnje prezentacijskog sloja u kojem Smarty vodi glavnu riječ.

```
$home->display('index.tpl');
```

index.tpl

- c. U memoriji poslužitelja ugrađuju se elementi XHTML-a:


```
{* smarty *}
```

- a. Učitava se dokument web_site.conf u kojem se nalaze varijable aplikacije koje se ne mijenjaju dinamički, već ih vlasnik ili autor aplikacije ručno upisuju:

```
{config_load file="web_site.conf"}
```

- b. Nastavlja se učitavanje XHTML-a:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>{#naslov#}</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../stil.css" type="text/css" />
</head>
<body>
```

- c. Učitava se datoteka zaglavlje.tpl:

```
{include file="zaglavlje.tpl"}
```

- d. Poziva se predložak knjiga_gostiju.tpl.

U ovoj točki procesuiranja naše aplikacije pogledat ćemo što se sve dogodilo u pozadini ovog poziva:

```
{include file="knjiga_gostiju.tpl"}
</body>
</html>
```

Poziv {include file="knjiga_gostiju.tpl"} pokrenuo je procesuiranje prezentacijskog sloja

U tijeku je procesuiranje svih Smarty predložaka: index.tpl, zaglavlje.tpl, knjiga_gostiju.tpl i function.load_knjiga_gostiju.php.

Za nas je novo i zanimljivo procesuiranje function.load_knjiga_gostiju.php i knjiga_gostiju.tpl.

function.load_knjiga_gostiju.php

- a. Evo funkcije koja kreira varijable za Smarty predložak i koja je pozvana i procesuirana se:

```
<?php
function smarty_function_load_knjiga_gostiju($params,
$smarty)
{
```

- b. Instancira se novi objekt KnjigaGostiju(). To je drugi kreirani objekt u cijeloj aplikaciji. Instanca objekta se dodjeljuje varijabli \$knjiga_gostiju. Klasa KnjigaGostiju s metodom init() će se učitati u memoriju uskoro (pogledajte niže).

```
$knjiga_gostiju = new KnjigaGostiju();
```

- c. Poziv svojoj metodi init():

```
$knjiga_gostiju->init();
```

- d. Poziv metodi ASSIGN() koja je u ranijim vježbama bila pozivana na stranici index.php, a sad je ta metoda dobila svoj poseban php dokument čije ime uvijek počinje s function.load_. To je zato što sad sadržaj dolazi iz baze u obliku asocijativnog arraya. To je tablica sa stupcima. Svaki stupac je poseban array. Svaki array se može ugraditi na različita mjesta u XHTML-u. Taj posao se obavlja s dvije datoteke: knjiga_gostiju.tpl i function.load_knjiga_gostiju.php. Nalazimo se u procesuiranju ove druge:

```
$smarty->assign($params['assign'], $knjiga_gostiju);
}
```

- e. Počinje učitavanje klase KnjigaGostiju. Sve što je programirano u ovoj klasi, konkretizirat će se u objektu koji je već u RAM-u instanciran.

```
class KnjigaGostiju
{
```

- f. Klasi treba ovaj javni član:

```
public $mKnjigaGostiju;
```

- g. i ovaj privatni član:

```
private $mPoslovniObjekt;
```

- h. Klasa ima metodu construct() koja se izvodi automatski prilikom instanciranja objekta.

```
function __construct()
{
```

- i. Automatski čim se instancira objekt KnjigaGostiju, instancira se i novi PoslovniObjekt. To je TREĆI objekt po redoslijedu nastanka u procesuiranju naše aplikacije.

Instanciranjem tog objekta dogodilo se i INICIJALIZIRANJE jer ima metodu construct() koja ga je automatski inicijalizirala, tj. metoda se automatski izvela.

Novi objekt se dodjeljuje privatnom članu mPoslovniObjekt. Slovo 'm' ispred imena znači 'na razini modula' tj. privatna varijabla klase. Klasa se nekad nazivala i 'modul'.

```
    $this->mPoslovniObjekt = new PoslovniObjekt();
}
```

```
function init()
```

- a. Metoda init() je pozvana na izvršavanje. Preko ove metode se prenosi asocijativni array od podatkovnog sloja, preko poslovnog sloja do prezentacijskog sloja u kojem se upravo nalazimo.

Ovog trenutka postoje tri objekta, tri konkretnosti. To su objekti BazaPodataka, KnjigaGostiju i PoslovniObjekt. Zadnja dva su instancirana i inicijalizirana upravo u ovom dokumentu.

Svom javnom članu mKnjigaGostiju, objekt dodijeljuje asocijativni array. Desna strana znaka jednakosti je poziv funkciji GetKnjigaGostiju() u PoslovnomObjektu.

```
{
    $this->mKnjigaGostiju = $this->mPoslovniObjekt-
>GetKnjigaGostiju();
}
}
?>
```

Ovaj poziv:

= \$this->mPoslovniObjekt->GetKnjigaGostiju();

Vodi nas do PoslovnogObjekta i njegove metode GetKnjigaGostiju().

Podsjetimo se kôda te metode:

```
public function GetKnjigaGostiju()
{
    $result = $this->mPodatkovniObjekt->GetKnjigaGostiju();
    return $result;
}
```

Pogledajmo u cijelosti klasu PoslovniObjekt da bi se prisjetili što se događa:

poslovni_objekt.php

```
<?php
```

- a. Učitava se klasa PodatkovniObjekt:

```
require_once SITE_ROOT .
'/podatkovni_objekti/podatkovni_objekt.php';
```

- b. Početak klase PoslovniObjekt

```
class PoslovniObjekt
{
```

- c. Klasa ima jednog privatnog člana u kojeg će pohraniti ono što dobije od PodatkovnogObjekta:

```
private $mPodatkovniObjekt;
```

- d. Počinje metoda construct():

```
function __construct()
{
```

- e. Svom jedinom članu dodijeljuje novo INSTANCIRANI PodatkovniObjekt. To je ČETVRTI objekt instanciran do ovog trenutka u našoj aplikaciji. Sjetimo se sva četiri redom kojim su nastajali: BazaPodataka, KnjigaGostiju, PoslovniObjekt, PodatkovniObjekt:

```
$this->mPodatkovniObjekt = new PodatkovniObjekt();
}
```

- f. PoslovniObjekt ima metodu GetKnjigaGostiju() čiji zadatak je vrlo, vrlo lagan. Ona samo poziva PodatkovniObjekt i njegovu metodu imenjakinju GetKnjigaGostiju().

```
public function GetKnjigaGostiju()
{
    $result = $this->mPodatkovniObjekt->GetKnjigaGostiju();
```

- g. Na ovom mjestu aplikacije već su prošlost ovi događaji: pozvan objekt PodatkovniObjekt, pozvana njegova metoda GetKnjigaGostiju():

podatkovni_objekti.php

```
<?php
class PodatkovniObjekt
{
    function __construct()
    {
        $this->bPodataka = $GLOBALS['gBazaPodataka'];
    }
}
```

- h. Metoda GetKnjigaGostiju():

```
public function GetKnjigaGostiju()
{
```

- i. kreira upit:

```
    $query_string = "SELECT id, Ime, Datum, Komentar FROM
knjigagostiju";
```

- j. Poziva objekt BazaPodataka (bPodataka) i njegovu metodu BpGetAll() i kao parametar šalje SQL upit. Objekt BazaPodataka drži otvorenu vezu prema bazi 'knjigagostiju' i sad pomoću svoje metode getAll() i upita kojeg je dobio preko svog parametra, pristupa bazi podataka i realizira upit. Rezultat toga je asocijativni array.

```
    $result = $this->bPodataka->BpGetAll($query_string);
```

- k. Šalje asocijativni array. Kome šalje? Šalje onom tko je pozvao metodu, a to je PoslovniObjekt. Ako na ovom mjestu u kôdu napišete:

```
    echo $result; exit;
```

i ponovo pozovete aplikaciju u web pregledniku, izbacit će na web stranici:

Array

Taj asocijativni array je ovo u našem konkretnom slučaju:

```
array(
array('id'=>'1', 'Ime'=>'Vinko', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'2', 'Ime'=>'Lucija', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'3', 'Ime'=>'Boris', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'4', 'Ime'=>'Iva', 'Datum'=>'2006-08-14', 'Komentar'=>'...')
)

    return $result;
}
}
?>
```

poslovni_objekt.php

```

public function GetKnjigaGostiju()
{
    $result = $this->mPodatkovniObjekt->GetKnjigaGostiju();

    a. Mi se nalazimo na ovom mjestu procesuiranja. Ako bismo ovdje
       napisali ovu instrukciju:
       echo $result; exit;
       aplikacija bi u web pregledniku izbacila 'Array'.
       Točno, na ovom mjestu imamo ovaj asocijativni array, pristigao
       iz podatkovnog sloja:

array(
array('id'=>'1', 'Ime'=>'Vinko', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'2', 'Ime'=>'Lucija', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'3', 'Ime'=>'Boris', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'4', 'Ime'=>'Iva', 'Datum'=>'2006-08-14', 'Komentar'=>'...')
)

    b. Naredba RETURN će taj array vratiti onom tko je pozvao
       PoslovniObjekt. Tko je pozvao? Prisjetimo se. Pozvao je objekt
       KnjigaGostiju koji se nalazi u prezentacijskom sloju.

return $result;

    c. RETURN vraća rezultat objektu KnjigaGostiju. Pa nastavimo
       gdje smo stali u objektu KnjigaGostiju.

}
}

```

function.load_knjiga_gostiju.php – NASTAVAK

Nalazimo se iza instrukcije koja je istaknuta boldiranim tekstom.

```

<?php
function smarty_function_load_knjiga_gostiju($params,
$smarty)
{
    $knjiga_gostiju = new KnjigaGostiju();
    $knjiga_gostiju->init();
    $smarty->assign($params['assign'], $knjiga_gostiju);
}
class KnjigaGostiju
{
    public $mKnjigaGostiju;
    private $mPoslovniObjekt;
    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();
    }
    function init()
    {
        $this->mKnjigaGostiju = $this->mPoslovniObjekt-
        >GetKnjigaGostiju();
    }
}

```

?>

- a. Upravo je PoslovniObjekt vratio objektu KnjigaGostiju ovaj asocijativni array:

```
array(
array('id'=>'1', 'Ime'=>'Vinko', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'2', 'Ime'=>'Lucija', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'3', 'Ime'=>'Boris', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'4', 'Ime'=>'Iva', 'Datum'=>'2006-08-14', 'Komentar'=>'...')
)
```

- b. Objekt KnjigaGostiju ima jednog javnog člana. To je \$mKnjigaGostiju. U tom članu je u memoriji poslužitelja pohranjen pristigli asocijativni array.
A sad sljedeći procesuiranje Smarty predložka knjiga_gostiju.tpl

knjiga_gostiju.tpl

- a. Ovo je komentar:

```
{* knjiga_gostiju.tpl *}
```

- b. Smarty se spaja na rezultate procesuiranja dokumenta function.load_knjiga_gostiju.php. U istom retku pozivamo metodu ASSIGN i dajemo ime knjiga_gostiju varijabli u koju će biti pohranjeni podaci. Varijabli će biti dodijeljeni (ASSIGN) podaci iz asocijativnog arraya.

```
{load_knjiga_gostiju assign="knjiga_gostiju"}
```

- c. U ovoj točki procesuiranja, Smarty ima pristup varijabli mKnjigaGostiju koja je u RAM-u. Smarty uzima u obradu varijablu mKnjigaGostiju čija je vrijednost asocijativni array:

```
array(
array('id'=>'1', 'Ime'=>'Vinko', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'2', 'Ime'=>'Lucija', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'3', 'Ime'=>'Boris', 'Datum'=>'2006-08-14', 'Komentar'=>'...'),
array('id'=>'4', 'Ime'=>'Iva', 'Datum'=>'2006-08-14', 'Komentar'=>'...')
)
```

- d. Dohvat podataka iz asocijativnog arraya se uvijek obavlja pomoću petlje. Smarty ima svoju sintaksu za petlju. Petlja se događa između ključnih riječi SECTION. SECTION ima atribut name i loop. Atribut loop definira da će se podaci iz varijable mKnjigaGostiju dodijeliti varijabli \$knjiga_gostiju:

```
{section name=i loop=$knjiga_gostiju->mKnjigaGostiju}
```

- e. U ovom retku smo pomoću Smartya dodali današnji datum i formatirali datum:

```
{$smarty.now|date_format:"%d.%m.%Y."}
```

- f. Evo sintakse pomoću koje petlja izvlači iz asocijativnog arraya, četiri obična arraya a to su STUPCI naše tablice: id, Ime, Datum, Komentar. Ujedno datum i formatiramo.

```

        {$knjiga_gostiju->mKnjigaGostiju[i].id}
        {$knjiga_gostiju->mKnjigaGostiju[i].Ime}
        {$knjiga_gostiju-
mKnjigaGostiju[i].Datum|date_format:"%d.%m.%Y."}
        {$knjiga_gostiju->mKnjigaGostiju[i].Komentar}
    {/section}

```

Smarty je napravio predloške i pohranio ih u mapu templates_c:

index.tpl.php

zaglavlje.tpl.php

knjiga_gostiju.tpl.php

To su sad gotovi PHP dokumenti. Ukoliko ne bude promjena u kôdu, sljedeći put će Smarty poslati te gotove predloške, pa će procesuiranje kraće trajati. To je veliki plus Smartyu.

Gdje smo stigli?

Stranica index.php je kao monitor nekog dizala koje se spušta i na kojem vidimo gdje se dizalo nalazi.

Boldirani tekst označava gdje se nalazimo!

>

index.php

```

<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>

```

dno_aplikacije.php

```

<?php
$GLOBALS['gBazaPodataka']->BpDisconnect();
?>

```

Globalna varijabla gBazaPodataka u kojoj je pohranjen objekt BazaPodataka, poziva svoju metodu za prekid veze s bazom!

I konačno je procesor došao do krajnjeg graničnika php programa:

```

?>

```

Nakon ovog znaka ?> kraja aplikacije, u RAM-u poslužitelja imamo konačni rezultat procesuiranja naše aplikacije. Taj rezultat je XHTML dokument:

```

<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>Knjiga gostiju</title>

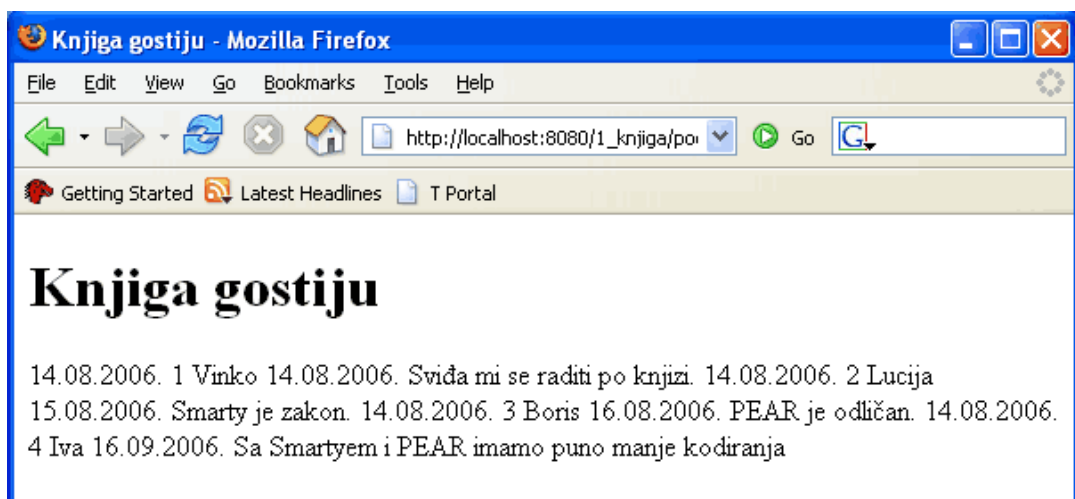
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2" />
<link rel="stylesheet" href="../stil.css" type="text/css" />
</head>
<body>
<h1>Knjiga gostiju</h1>
```

```
14.08.2006.
    1
    Vinko
    14.08.2006.
    Sviđa mi se raditi po knjizi.
14.08.2006.
    2
    Lucija
    15.08.2006.
    Smarty je zakon.
14.08.2006.
    3
    Boris
    16.08.2006.
    PEAR je odličan.
14.08.2006.
    4
    Iva
    16.09.2006.
    Sa Smartyem i PEAR imamo puno manje kodiranja
</body>
</html>
```

I što procesor tad učini? Procesor preda web poslužitelju Apache taj XHTML dokument i OBRIŠE sve što je ova aplikacija pohranila u RAM-u. Sve!

Apache pošalje XHTML dokument našem web klijentu IE ili FireFoxu ili Operi:



Ako ponovo zatražimo istu stranicu, sve što smo opisali će se ponovo dešavati. Apache će prihvatiti naš novi poziv kao bilo čiji drugi. On ne zna da smo mi maloprije već učitali tu istu stranicu.

Kraj procesuiranja

Sve instrukcije naše aplikacije na jednom mjestu

Prošli smo kroz kompletan kôd aplikacije Knjiga gostiju. Prikažimo sad u nizu kako su učitali u RAM poslužitelja, sve instrukcije naše aplikacije.

```
<?php
require_once 'include/vrh_aplikacije.php';

<?php
require_once 'konfiguracija.inc.php';

<?php
define("SITE_ROOT", dirname(dirname(__FILE__)));
define("SMARTY_DIR", SITE_ROOT."/biblioteke/smarty/");
define("TEMPLATE_DIR", SITE_ROOT."/templates/");
define("COMPILE_DIR", SITE_ROOT."/templates_c/");
define("CONFIG_DIR", SITE_ROOT."/configs/");

if ((substr(strtoupper(PHP_OS), 0, 3)) == "WIN")
    define("PATH_SEPARATOR", ";");
else
    define("PATH_SEPARATOR", ":");
ini_set('include_path', SITE_ROOT . '/biblioteke/PEAR' .
PATH_SEPARATOR . ini_get('include_path'));
//podaci za logiranje u bazu
define("USE_PERSISTENT_CONNECTIONS", "true");
define("DB_SERVER", "localhost");
define("DB_USERNAME", "admin");
define("DB_PASSWORD", "admin");
define("DB_DATABASE", "knjigagostiju");
define("MYSQL_CONNECTION_STRING", "mysql://" . DB_USERNAME .
":" .
        DB_PASSWORD . "@" . DB_SERVER . "/" . DB_DATABASE);
?>

require_once 'uspostavljanje_smarty.php';

<?php
require_once SMARTY_DIR . 'Smarty.class.php';
class HomePage extends Smarty
{
function __construct()
{
$this->Smarty();
$this->template_dir = TEMPLATE_DIR;
$this->compile_dir = COMPILE_DIR;
$this->config_dir = CONFIG_DIR;

$this->plugins_dir[0] = SMARTY_DIR . 'plugins';
$this->plugins_dir[1] = SITE_ROOT . "/smarty_plugins";
}
```

```
}
?>

require_once 'bazapodataka.php';

<?php
require_once 'DB.php';
class BazaPodataka
{
    public $bp;
    function __construct($connectionString)
    {
        $this->bp = DB::connect($connectionString,
                                USE_PERSISTENT_CONNECTIONS);
        $this->bp->setFetchMode(DB_FETCHMODE_ASSOC);
    }
    public function BpDisconnect()
    {
        $this->bp->disconnect();
    }
    public function BpQuery($queryString)
    {
        $result = $this->bp->query($queryString);
        return $result;
    }
    public function BpGetAll($queryString)
    {
        $result = $this->bp->getAll($queryString);
        return $result;
    }
    public function BpGetRow($queryString)
    {
        $result = $this->bp->getRow($queryString);
        return $result;
    }
    public function BpGetOne($queryString)
    {
        $result = $this->bp->getOne($queryString);
        return $result;
    }
}
?>

//Instanciranje globalnog objekta BazaPodataka
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);

?>

require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';

<?php
```

```

require_once SITE_ROOT .
'/podatkovni_objekti/podatkovni_objekt.php';

<?php
class PodatkovniObjekt
{
    function __construct()
    {
        $this->bPodataka = $GLOBALS['gBazaPodataka'];
    }
    public function GetKnjigaGostiju()
    {
        $query_string = "SELECT id, Ime, Datum, Komentar FROM
knjigagostiju";
        $result = $this->bPodataka->BpGetAll($query_string);
        return $result;
    }
}
?>

class PoslovniObjekt
{
    private $mPodatkovniObjekt;
    function __construct()
    {
        $this->mPodatkovniObjekt = new PodatkovniObjekt();
    }
    public function GetKnjigaGostiju()
    {
        $result = $this->mPodatkovniObjekt->GetKnjigaGostiju();

        // echo $result; exit;
        return $result;
    }
}
?>

$home = new HomePage();
$home->display('index.tpl');

{* smarty *}
{config_load file="web_site.conf"}
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>{#naslov#}</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-2" />
<link rel="stylesheet" href="../../../stil.css" type="text/css" />
</head>

```

```

<body>
{include file="zaglavlje.tpl"}

<h1>Knjiga gostiju</h1>

{include file="knjiga_gostiju.tpl"}

<?php
function smarty_function_load_knjiga_gostiju($params,
$smartyy)
{
    $knjiga_gostiju = new KnjigaGostiju();
    $knjiga_gostiju->init();
    $smarty->assign($params['assign'], $knjiga_gostiju);
}
class KnjigaGostiju
{
    public $mKnjigaGostiju;
    private $mPoslovniObjekt;
    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();
    }
    function init()
    {
        $this->mKnjigaGostiju = $this->mPoslovniObjekt-
>GetKnjigaGostiju();
    }
}
?>

{* knjiga_gostiju.tpl *}
{load_knjiga_gostiju assign="knjiga_gostiju"}
{section name=i loop=$knjiga_gostiju->mKnjigaGostiju}
{$smarty.now|date_format:"%d.%m.%Y."}
    {$knjiga_gostiju->mKnjigaGostiju[i].id}
    {$knjiga_gostiju->mKnjigaGostiju[i].Ime}
    {$knjiga_gostiju-
>mKnjigaGostiju[i].Datum|date_format:"%d.%m.%Y."}
    {$knjiga_gostiju->mKnjigaGostiju[i].Komentar}
{/section}

</body>
</html>

require_once 'include/dno_aplikacije.php';
?>

<?php
$GLOBALS['gBazaPodataka']->BpDisconnect();
?>

```

KRAJ

Runtime aplikacije

Runtime je vrijeme koje aplikacija provede u memoriji poslužitelja. Preciznije rečeno to vrijeme se mjeri od trenutka kad procesor uzme prvu instrukciju aplikacije pa do trenutka kad završi obradu zadnje instrukcije iste aplikacije.

Kako obradi pojedinu instrukciju, tako procesor uništi tu instrukciju, a rezultat obrade te instrukcije pohrani u RAM poslužitelja.

Klase procesor pohrani takve kakve su u RAM poslužitelja. Ne procesuiraju njihov kôd. To su mrtve kopije u ladicama RAM-a. Zato su klase apstrakcija, apstraktni podaci.

Praktično, posao za koji je namijenjena aplikacija počinje s prvom instancom objekta. Objekt je konkretnost klase. I sad se izvršavaju naredbe koje su bile mrtve i neaktivne u klasi. Ne možemo instancirati objekt ako nema njegove klase.

Iz toga proizlazi: u aplikaciji najprije učitamo potrebne klase, a onda instanciramo objekte.

Sažetak runtimea aplikacije

Ova aplikacija je učinila samo jedan posao: donijela je na web stranicu sve retke tablice 'knjigagostiju' iz istoimene baze podataka. 100% je objektno orijentirana i razdvojeni su XHTML i PHP, pa web kreator i programer mogu raditi potpuno odvojeno i neovisno jedan o drugom.

Brojke aplikacije:

1. Ima 112 instrukcija.
2. Ima 5 naših klasa plus klase paketa SMARTY i PEAR.
3. Sve su klase konkretizirane, znači ni jedna nije suvišna.

Redoslijed učitavanja (procesuiranja):

1. Klasa Smarty.
2. HomePage, klasa nasljednica klase Smarty.
3. PEAR Klasa DB.
4. Klasa BazaPodataka.
 - a. INSTANCIRA se PRVI OBJEKT BazaPodataka. Taj objekt obavi posao spajanja na bazu podataka i taj spoj ostaje aktivan do kraja aplikacije.
5. Klasa PodatkovniObjekt
6. Klasa PoslovniObjekt
 - a. INSTANCIRA se DRUGI OBJEKT HomePage
 - b. Objekt HomePage pozove metodu DISPLAY("index.tpl")
7. Klasa KnjigaGostiju

- a. index.tpl poziva na učitavanje knjiga_gostiju.tpl i to aktivira proces koji sljedeći:
 - b. INSTANCIRA se TREĆI OBJEKT KnjigaGostiju
 - c. INSTANCIRA se ČETVRTI OBJEKT PoslovniObjekt
 - d. INSTANCIRA se PETI OBJEKT PodatkovniObjekt
8. Globalni objekt BazaPodataka koji je memorirao spoj na bazu se samorazara. Procesor uništava sve u memoriji, što je povezano na bilo koji način s našom aplikacijom.

Koliko je u realnom vremenu trajao runtime?

Potpuno jednoznačno pitanju u naslovu je pitanje: "Koliko je živio naš program?".

Za stvoriti sebi sliku realnog vremena života naše aplikacije, možemo i ovako razmišljati. Ako je poslužitelj računalo s procesorom 3GHz, to znači da u jednoj sekundi izvrši 3 milijarde taktova. Ako u jednom taktu izvrši jednu našu instrukciju, onda je naša aplikacija živjela 112 taktova. Složit ćete se, vrlo kratak život u realnom vremenu imaju naši objekti.

A sad ispričajmo u kratko što su radili naši objekti

1. INSTANCIRA se PRVI OBJEKT BazaPodataka


```
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
```

 i automatski pomoću svoje metode construct() napravi sljedeće:
 - a. Prenosi konstantu MYSQL_CONNECTION_STRING u metodu construct().
 - b. Pomoću podataka u konstanti i PEAR objekta DB, naš se objekt spaja na bazu podataka.
 - c. Spoj ostaje na snazi do daljnjeg jer smo USE_PERSISTENT_CONNECTIONS stavili na 'true'.
 - d. Angažiramo metodu setFetchMode(DB_FETCHMODE_ASSOC) i tako određujemo da nam PEAR objekt DB vraća ASOCIJATIVNI ARRAY.
 - e. Sve se to pohranjuje u globalnu varijablu \$gBazaPodataka.
2. INSTANCIRA se DRUGI OBJEKT HomePage koji posjeduje sva svojstva i sve metode klase Smarty.
 - f. Objekt HomePage pozove metodu DISPLAY("index.tpl")
 - g. index.tpl poziva na učitavanje knjiga_gostiju.tpl i to aktivira proces koji sljedeći:
3. INSTANCIRA se TREĆI OBJEKT KnjigaGostiju.
 - a. Automatski instancira Poslovni objekt.
 - b. Poziva metodu GetKnjigaGostiju() PoslovnogObjekta.

4. INSTANCIRA se ČETVRTI OBJEKT PoslovniObjekt
 - a. Automatski instancira PodatkovniObjekt.
 - b. Metoda GetKnjigaGostiju() je pozvana, pa sad ta metoda poziva istoimenu metodu u PodatkovnomObjektu.
5. INSTANCIRA se PETI OBJEKT PodatkovniObjekt
 - a. Pomoću globalnog objekta BazaPodataka spaja se na bazu podataka.
 - b. Izvršava se pozvana metoda GetKnjigaGostiju() koja šalje upit bazi:


```
public function GetKnjigaGostiju()
{
    $query_string = "SELECT id, Ime, Datum, Komentar FROM
knjigagostiju";
    $result = $this->bPodataka->BpGetAll($query_string);
    return $result;
}
```
 - c. Baza vraća PodatkovnomObjektu asocijativni array. Podatkovni objekt vraća taj asocijativni array PoslovnomObjektu. PoslovniObjekt vraća asocijativni array objektu KnjigaGostiju. Objekt KnjigaGostiju predaje asocijativni array Smartyu. Smarty ugrađuje asocijativni array u XHTML naše web stranice. Naša web stranica je napravljena i web poslužitelj Apache je šalje našem web klijentu.
6. Globalni objekt BazaPodataka koji je memorirao spoj na bazu se samorazara. Procesor uništava sve u memoriji, što je povezano na bilo koji način s našom aplikacijom.

Još jedan pogled na cijeli naš program - index.php

```
<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>
```

Cijeli naš program se nalazi u ovih pet instrukcija. Kad pozovemo adresu:

http://localhost:8080/1_knjiga/poglavlje_knjigagostiju/k1/

mi smo pozvali index.php. Datoteke su fizički odvojene. Svaka za sebe je veći ili manji program. Zajedno čine neki posao. Konkretno u ovom slučaju taj posao je izgradnja dinamičkog sadržaja i cijele web stranice index.php.

Programeri najčešće pišu svaku klasu u zasebne datoteke. To bi bila osobna biblioteka klasa. Svaku klasu programer koristi za izgradnju novih aplikacija. Dobro organiziran programer, s povećom bibliotekom klasa, može vrlo brzo napraviti vrhunsku profesionalnu aplikaciju.

Svaki program funkcionira tako što objekti pozivaju druge objekte, ovi treće, pa onda taj treći nešto napravi i rezultat vrati drugom objektu itd. Vrlo živa komunikacija između objekata. To je B2B komunikacija. Svaka klasa i njen objekt, fizički mogu biti pohranjeni na bilo kojem poslužitelju na mreži. Oni će pozvati jedni druge i napraviti posao. Svaka klasa i njen objekt su business za sebe. Neka klasa može biti pozivana od više različitih aplikacija. Po želji mnogo. Svaka klasa radi svoj posao. Od tud naziv B2B (business to business).

Ostale vježbe

Ostale vježbe vezane uz 'Knjigu gostiju' bave se XHTML-om i CSS-om, što nije tema ovog obrazovnog sadržaja.

Što slijedi

S ovom posljednjom sažetom pričom o tome što rade naši objekti u web aplikaciji, započnimo našu sljedeću aplikaciju koja nosi općenito ime CMS (Content Management System).

Prva vježba našeg CMS-a bit će kopija ove vježbe. Jer, u našem CMS-u pozivamo na web stranicu tri razine sadržaja: područja, cjeline, jedinice. Područja se pojavljuju odmah na HomePage, a to je ono što smo imali i u ovoj vježbi.

Područja su glavni izbornici.

Tad postaje zanimljivo i novo za nas. Kako smo s klik na neko područje učitali cjeline tog područja.

A onda na isti način riješavamo sljedeći klik na neku cjelinu, kad se učitavaju sve jedinice te cijeline. Pošto jedinica može biti mnogo, učitavamo ih četiri po četiri s linkovima sljedeće, prethodne.

I još puno novih detalja.

Sretno u novom poglavlju.



CMS

Metodika-----

I evo nam jedne izvanredne web aplikacije. Za što? Za sve. Za vas osobno, kao osobni web site, za vasu školu kao školski portal, za učenike i njihovu praksu. Ovu aplikaciju možete pokazivati u radu prije nego ste napisali s učenicima iti jedan kôd. Da vide što će napraviti. Da vide što je cilj programiranja u ovoj knjizi, u ovoj školskoj godini. Isto tako, vidjet će što znači administrirati web site preko Web preglednika. Bez FTP-a, bez FrontPage-a. Od bilo kud. Iz Internet cafee, kuće, škole.

Ako su učenici prošli XHTML i CSS, mogu početi redizajnirati CMS za sebe odmah, bez da znaju išta o PHP. Tako će naučiti kako je lako redizajnirati ovu aplikaciju u kojoj je Smarty odvojio XHTML od PHP.

CMS je primjena svega naučenog do sad. Primjena elementarnog znanja. Aplikacija CMS Ivamar je ovog trenutka savršeno projektirana web aplikacija:

- 100% OOP.
- U tri sloja.
- Smarty razdvaja XHTML od PHP.
- PEAR spoj na bazu podataka.

Uz objašnjenja koja sljede, ako nešto do sad nije bilo jasno, ovdje je prilika za razjašnjenje.

Ponavljam: moj cilj s ovom knjigom je reći ono što nećete naći na Internetu. Ono što je u mojoj glavi. Što je mene zanimalo kod rada programa i kod učenja programiranja. Zato, ako naidete na nešto što ova knjiga ne pokriva dovoljno, ili onako kako vama treba, ne brinite, tipkajte to u Google i sve što vas zanima ćete dobiti.

Udžbenik-----

Što ćemo naučiti?

1. Što je CMS?
2. Administriranje CMS-a.
3. Koristit ćemo znanje iz "Knjige gostiju" za početak rada na CMS-u.
4. Programirati u objektno orijentiranom okruženju.

5. Aplikacija u tri sloja: prezentacijski, poslovni i podatkovni sloj.
6. Programirati uz pomoć paketa Smarty i PEAR.

Što je CMS

CMS je skraćenica od content management system ili u prijevodu sustav za upravljanje sadržajem. To je općenit naziv, i u hijerarhiji tipova web siteova, to je krovni web sitea. Svi drugi tipovi web aplikacija su samo specijalizirana podskupina CMS-a. Na primjer:

- E-novine
- E-foto galerije
- E-learning ili e-učenje
- Portali
- Školski web siteovi
- Osobni web siteovi
- Web siteovi firmi itd.
- E-trgovina do momenta kad počinje naplaćivanje
- Portali
- itd.

Najkraće rečeno, gotovo sve na Webu štim čovjek dolazi u kontakt, sve na Webu što ima grafičko sučelje za čovjeka, je neka vrsta CMS-a. CMS u ovoj knjizi možete primjeniti za sve gore navedene tipove web aplikacija, uz potrebu nikakve ili beznačajne promjene.

Jasno, grafičkom dizajnu neke web aplikacije, kao i funkcionalnostima koje ona pruža nema kraja, i ovaj CMS se može po želji unaprijediti. I to vrlo lako i brzo jer je programiran 100% u objektno orijentiranom okruženju i pomoću Smartya je odvojen XHTML od PHP-a. Aplikacija se spaja na bilo koju bazu pomoću PEAR-a. Sve informacije između prezentacijskog sloja i podatkovnog sloja prolaze kroz srednji, poslovni sloj.

U poslovni sloj uvijek možemo ugrađivati svoja željena 'pravila igre', bez potrebe za ikakvim promjenama u ostala dva sloja.

CMS ima dva dijela. Jedan dio vide posjetitelji, a drugi dio, administrativni, vidi samo autor. To su i dvije zasebne web aplikacije i obje su podjednako važne. Ova strana našeg CMS-a okrenuta prema posjetiteljima ima svoje ciljeve i publiku. Administrativna strana ima svoje ciljeve i svoju publiku. Publika administrativne strane je provjerena prije pristupa, jer su to ljudi koji svaki dan ažuriraju sadržaj. Sve više ljudi radi to 8 sati dnevno. To im je radno mjesto. Zato nije svejedno kako je taj administrativni dio organiziran i kako je dizajniran. O tome ovisi učinkovitost tih ljudi. A o tome ovisi i da li ćete ili nećete uspijeti prodati svoje web aplikacije.

Za one koji ovdje počinju raditi po knjizi

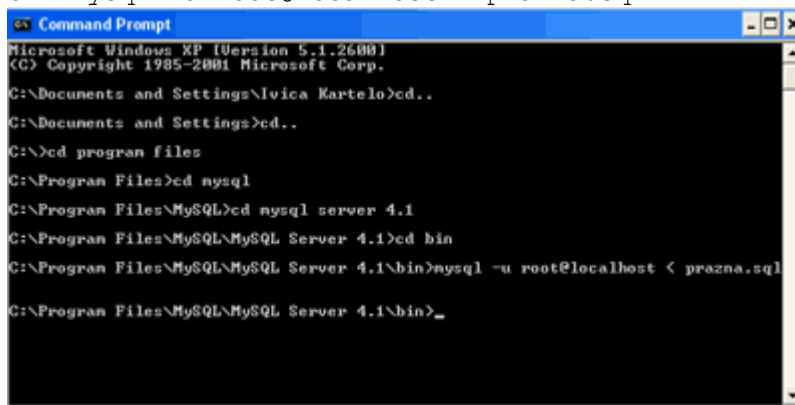
Ukoliko ste išli odmah na ovo poglavlje, a nemate na svom računalu instaliran PHP, učinite sljedeće. Uz knjigu su i svi primjeri i sav potreban software. Pomoću XAMPP izvršne datoteke instalirajte svav potreban software. Na sva pitanja tijekom instalacije odgovarajte potvrdno.

Na kraju instalacije ćete na svom računalu imati Apache, PHP, MySQL. Možete ići na sljedeći naslov u knjizi.

CMS – instalacija

1. Sadržaj mape cms_komplet kopirajte u mapu na svom računalu. Toj mapi mora pristupati Apache. Najjednostavnije je kopirati mapu 'cms_komplet' u mapu 'htdocs' (na mom računalu put je: C:\Program Files\Apache Group\Apache2\htdocs) pa će url aplikacije biti http://localhost:8080/cms_komplet/
2. Kreirajte MySQL bazu podataka ovako:
 - a. Datoteku \cms_komplet\baza_podataka\prazna.sql kopirajte u MySQL-ovu mapu 'bin' čiji je put na mom računalu ovaj:
C:\Program Files\MySQL\MySQL Server 4.1\bin
3. Otvorite Command Prompt (Start/Run/cmd/Enter).
4. Dodite do MySQL-ove mape bin, u mom slučaju je to:
C:\Program Files\MySQL\MySQL Server 4.1\bin
5. U naredbodavni redak bin> tipkajte ovo:

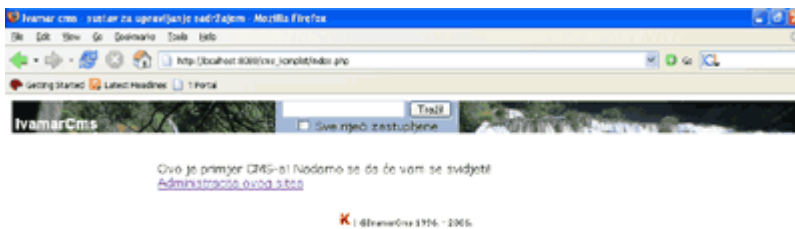
```
bin>mysql -u root < prazna.sql  
ili  
bin>mysql -u root@localhost < prazna.sql
```



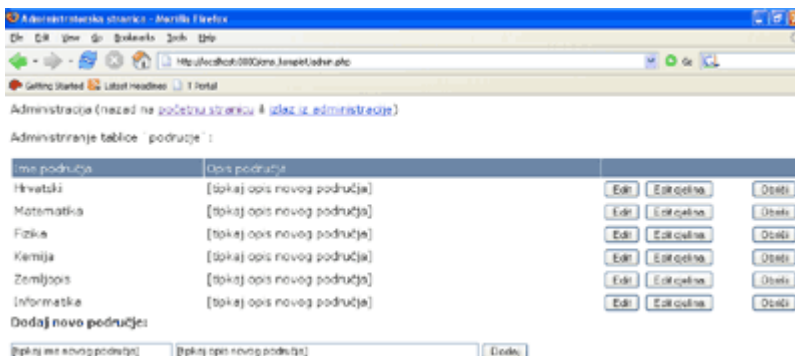
6. Enter. Baza po imenu 'prazna' je instalirana.

CMS - administriranje

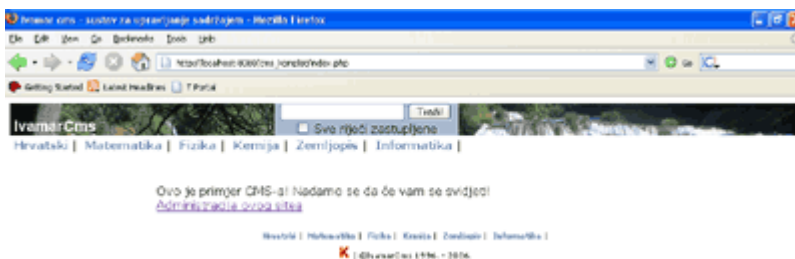
1. Otvorite Web preglednik i tipkajte adresu:
http://localhost:8080/cms_komplet/



2. Klik na link Administracija ovog sitea.
3. Korisničko ime: admin
4. Zaporka: admin
5. Dodaj novo područje: tipkajte Hrvatski, pa klik na Dodaj.
6. Dodaj novo područje: tipkajte Matematika, pa klik na Dodaj.
7. Dodaj novo područje: tipkajte Fizika, pa klik na Dodaj.
8. Dodaj novo područje: tipkajte Kemija, pa klik na Dodaj.
9. Dodaj novo područje: tipkajte Zemljopis, pa klik na Dodaj.
10. Dodaj novo područje: tipkajte Informatika, pa klik na Dodaj.



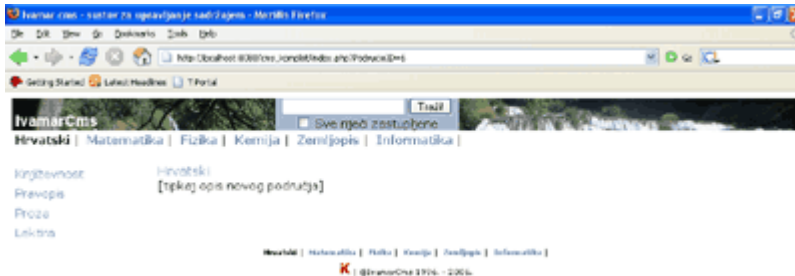
11. Klik na link Nazad na početnu stranicu:



Dobili smo glavni vodoravni izbornik od upravo unesenih podataka.

12. Klik na IvamarCMS i dobit ćete HomePage na kojoj je link Administracija ovog sitea.
13. Klik na Administracija ovog sitea.
14. Klik na Edit cjelina u retku Hrvatski.

15. U polje Dodaj novu cjelinu tipkajte redom Književnost, Pravopis, Proza, Lektira i iza svake riječi klik na Dodaj.
16. Klik na link 'nazad na početnu stranicu'.
17. Klik na link Hrvatski. U okomitom izborniku su se pojavile riječi koje smo upravo unijeli. Hrvatski je područje, a ovo su cjeline tog područja.



18. Nastavite unijeti po volji cjeline za ostala područja, Matematika, Fizika itd.
19. Unesite i jedinice za pojedine cjeline, ovako:
 - a. Klik na IvamarCMS i dobit ćete HomePage na kojoj je link Administracija ovog sitea.
 - b. Klik na Administracija ovog sitea.
 - i. (admin, admin, ako je potrebno).
 - c. U retku Hrvatski odaberite gumb Edit cjelina.
 - d. U retku Književnost odaberite gumb Edit jedinica.
 - e. U polje dodaj novu jedinicu tipkajte Književnost 16. stoljeća pa odaberite gumb Dodaj.
 - f. U polje dodaj novu jedinicu tipkajte Književnost 17. stoljeća pa odaberite gumb Dodaj.
 - g. U polje dodaj novu jedinicu tipkajte Književnost 18. stoljeća pa odaberite gumb Dodaj.
 - h. U polje dodaj novu jedinicu tipkajte Književnost 19. stoljeća pa odaberite gumb Dodaj.
 - i. U polje dodaj novu jedinicu tipkajte Književnost 20. stoljeća pa odaberite gumb Dodaj.
 - j. U polje dodaj novu jedinicu tipkajte Književnost 21. stoljeća pa odaberite gumb Dodaj.
20. Sad odaberite redom gumb Edit za svaku jedinicu i u polje Opis tipkajte bilo koji tekst, može i nesuvisli za vježbu, pa klik na gumb Ažuriraj.
21. Odaberite link nazad na početnu stranicu.
22. Odaberite područje Hrvatski.
23. Odaberite cjelinu Književnost.

Pojavile su se 4 jedinice i linkovi ' Stranica 1 od ukupno 2: Prethodna Sljedeća'

Upload slika

1. Klik na IvamarCMS i dobit ćete HomePage na kojoj je link Administracija ovog sitea.
2. Klik na Administracija ovog sitea.
 - i. (admin, admin, ako je potrebno).
3. Klik na Edit cjelina u retku Hrvatski.
4. Klik na Edit jedinica u retku Književnost.
5. Klik na Odaberi u retku Književnost 16. stoljeća.
6. Klik u polje Slika 1, klik na Browse ako ste u FireFoxu, Pregledaj... ako ste u IE i imate lokalizirane Windowse na hrvatski, Choose ako ste u Operi.
 - a. Odaberite za vježbu bilo koju sliku, bilo gdje se nalazila lokalno na vašem računalu.
 - b. Klik na Objavi.
7. Klik u polje Slika 2, klik na Browse ako ste u FireFoxu, Pregledaj... ako ste u IE i imate lokalizirane Windowse na hrvatski, Choose ako ste u Operi.
 - a. Odaberite za vježbu bilo koju sliku, bilo gdje se nalazila lokalno na vašem računalu.
 - b. Klik na Objavi.
8. Klik na link 'nazad na listu svih jedinica u cjelini'.
9. Kliknite na gumb Odaberi u retku Književnost 17. stoljeća.
10. Ponovite korake 6. i 7.
11. Na gore opisani način objavite sliku za svaku jedinicu Književnost...
12. Odaberite link 'nazad na početnu stranicu'.
13. Odaberite Hrvatski, Književnost:



Promidžba

CMS ima mogućnost promidžbe JEDINICA na dvije razine:

1. Na Home Page. To je ono što će posjetitelj prvo ugledati kad dođe na naš sitea. U CMS-u se sadržaj cijelog web sitea izmjenjuje na jednoj stranici: index.php.
2. Na razini PODRUČJA.

Ako ste radili kao ja u knjizi, vi nemate još niti jednu jedinicu u promidžbi.

3. Unesite promidžbe za onih šest jedinica što smo ih unijeli. Za njih pet uključite opciju Promidžba na razini Home, a za četiri uključite Promidžba na razini područja, ovako:
 - a. Klik na IvamarCMS i dobit ćete HomePage na kojoj je link Administracija ovog sitea.
 - b. Klik na Administracija ovog sitea.
 - i. (admin, admin, ako je potrebno).
 - c. U retku Hrvatski odaberite gumb Edit cjelina.
 - d. U retku Književnost odaberite gumb Edit jedinica.
 - e. U retku Književnost 16. stoljeća odaberite gumb Edit.
 - f. Uključite opciju Promidžba na razini Home. Ažuriraj.
 - g. Nazad na cjeline.
 - h. Ponavljajte korake a., b. i c. dok ne dodijelite svim jedinicama željene Promidžbe.
4. Nazad na početnu stranicu i vidjet ćete JEDINICE KOJE PROMOVIRATE na razini POČETNE stranice.

5. Klik na Hrvatski i vidjet ćete JEDINICE KOJE PROMOVIRATE na razini PODRUČJA Hrvatski.

Napomena: Svaki redak u tablici JEDINICE predstavlja jednu web stranicu. Najveći sadržaj dolazi iz polja Opis. U praksi, kreiramo web stranicu (XHTML dokument) u nekom grafičkom editoru web stranica, npr. FrontPageu ili Dreamweaveru. To radimo off line, lokalno na svom računalu. Kad smo stranicu napravili, spojimo se na Internet i u polje Opis kopiramo XHTML dokument. One slike koje nisu Slika 1 i Slika 2 u našem CMS-u, objavimo preko svog FTP-a u mapu koju kreiramo na svom hosting-u.

Programirajmo CMS

Vježba 1

Vježba se nalazi u mapi cms1_kod.

Zadatak ove vježbe je sljedeći:

1. Prikaz baze podataka nad kojom radimo CMS.
2. Kreiraj sve potrebne mape kao u Vježbi 1 aplikacije Knjiga gostiju.

Baza podataka

prazna.sql

```
# Struktura tablice `podrucje`
#

DROP DATABASE IF EXISTS `prazna`;

CREATE DATABASE prazna;

GRANT ALL PRIVILEGES ON prazna.* TO admin@localhost
IDENTIFIED BY 'admin' WITH GRANT OPTION;

USE prazna;

DROP TABLE IF EXISTS `podrucje`;

CREATE TABLE `podrucje` (
  `podrucje_id` int(11) NOT NULL auto_increment,
  `ime` varchar(50) NOT NULL default '',
  `opis` TEXT default NULL,
  PRIMARY KEY (`podrucje_id`)
) TYPE=MyISAM AUTO_INCREMENT=5 ;

DROP TABLE IF EXISTS `cjelina`;

CREATE TABLE `cjelina` (
  `cjelina_id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `podrucje_id` INT UNSIGNED NOT NULL ,
```

```

ime` VARCHAR( 50 ) NOT NULL ,
`opis` TEXT ,
PRIMARY KEY ( `cjelina_id` )
) TYPE=MyISAM;

DROP TABLE IF EXISTS `jedinica`;

CREATE TABLE `jedinica` (
`jedinica_id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
ime` VARCHAR( 50 ) NOT NULL ,
`opis` TEXT NOT NULL ,
`slika_1` VARCHAR( 50 ) ,
`slika_2` VARCHAR( 50 ) ,
`promidzba_na_razini_homepage` VARCHAR( 1 ) NOT NULL ,
`promidzba_na_razini_podrucja` VARCHAR( 1 ) NOT NULL ,
PRIMARY KEY ( `jedinica_id` )
) TYPE=MyISAM;

ALTER TABLE `jedinica` ADD FULLTEXT ImeOpisFTIndex(ime,opis);

DROP TABLE IF EXISTS `cjelina_jedinica`;

CREATE TABLE `cjelina_jedinica` (
`cjelina_id` INT UNSIGNED NOT NULL ,
`jedinica_id` INT UNSIGNED NOT NULL ,
PRIMARY KEY ( `cjelina_id` , `jedinica_id` )
) TYPE=MyISAM;

```

Nakon što smo unijeli podatke naša baza izgleda ovako

Tablica: podrucje

podrucje_id	ime	opis
1	Hrvatski	tekst
2	Matematika	tekst
3	Fizika	tekst
4	Kemija	tekst
5	Zemljopis	tekst
6	Informatika	tekst

Tablica: cjelina

cjelina_id	podrucje_id	ime	opis
1	1	Književnost	tekst
2	1	Pravopis	tekst
3	1	Proza	tekst
4	1	Lektira	tekst

Tablica: cjelina_jedinica

cjelina_id	jedinica_id
1	1
1	2
1	3
1	4
1	5
1	6

Tablica: jedinica

jedinica_id	ime	opis	slika_1	slika_2	promidzba_na_razini_homepage	promidzba_na_razini_podrucja
1	Književnost 16. stoljeća	tekst			1	0
2	Književnost 17. stoljeća	tekst			1	0
3	Književnost 18. stoljeća	tekst			1	1
4	Književnost 19. stoljeća	tekst			1	1
5	Književnost 20. stoljeća	tekst			1	1
6	Književnost 21. stoljeća	tekst			0	1

Relacije među tablicama

Tablica: podrucje

podrucje_id	ime	opis
1	Hrvatski	tekst
2	Matematika	tekst
3	Fizika	tekst
4	Kemija	tekst
5	Zemljopis	tekst
6	Informatika	tekst

Tablica: cjelina

cjelina_id	podrucje_id	ime	opis
1	1	Književnost	tekst
2	1	Pravopis	tekst
3	1	Proza	tekst
4	1	Lektira	tekst

Tablica: cjelina_jedinica

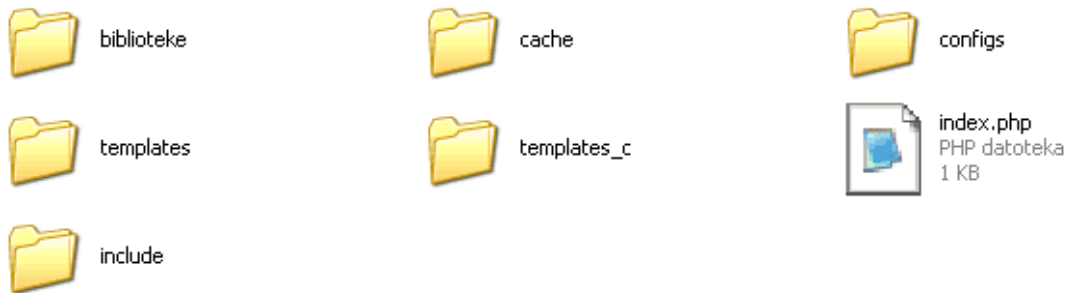
cjelina_id	jedinica_id
1	1
1	2
1	3
1	4
1	5
1	6

Tablica: jedinica

jedinica_id	ime	opis	slika_1	slika_2	promidzba_na_razini_homepage	promidzba_na_razini_podrucja
1	Književnost 16. stoljeća	tekst			1	0
2	Književnost 17. stoljeća	tekst			1	0
3	Književnost 18. stoljeća	tekst			1	1
4	Književnost 19. stoljeća	tekst			1	1
5	Književnost 20. stoljeća	tekst			1	1
6	Književnost 21. stoljeća	tekst			0	1

Rad

1. Kreirajte korijen mapu za vježbu 1, na primjer cms1_kod.
2. U mapi cms1_kod kreirajte ove mape:



3. U mapu 'biblioteke' kopiraj mapu 'smarty' iz ranijih vježbi.
4. U mapi cms1_kod kreirajte dokument

index.php:

```
<?php
require_once 'include/vrh_aplikacije.php';
$home = new HomePage();
$home->display('index.tpl');
?>
```

5. U mapi include kreirajte ove dokumente:

uspostavljanje_smarty.php

```
<?php
require_once SMARTY_DIR . 'Smarty.class.php';
class HomePage extends Smarty
{
function __construct()
{
$this->Smarty();
$this->template_dir = TEMPLATE_DIR;
$this->compile_dir = COMPILE_DIR;
$this->config_dir = CONFIG_DIR;
}
}
?>
```

vrh_aplikacije.php

```
<?php
require_once 'konfiguracija.inc.php';
require_once 'uspostavljanje_smarty.php';
?>
```

konfiguracija.inc.php

```
<?php
define("SITE_ROOT", dirname(dirname( FILE_ )));
define("SMARTY_DIR", SITE_ROOT."/biblioteke/smarty/");
define("TEMPLATE_DIR", SITE_ROOT."/templates/");
define("COMPILE_DIR", SITE_ROOT."/templates_c/");
define("CONFIG_DIR", SITE_ROOT."/configs/");
?>
```

6. U mapi configs kreirajte dokuemnt:

web_site.conf

```
naslov = "Upravljanje sadržajem"
```

7. U mapi templates kreirajte ove dokumente:

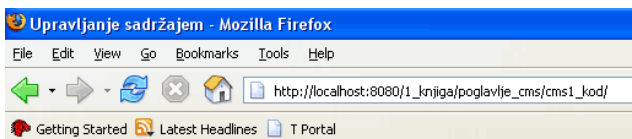
index.php

```
{* smarty *}
{config_load file="web_site.conf"}
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<title>{#naslov#}</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
<link rel="stylesheet" href="../../stil.css" type="text/css" />
</head>
<body>
{include file="zaglavlje.tpl"}
</body>
</html>
```

zaglavlje.php

```
<h1>Upravljanje sadržajem</h1>
```

8. Otvorite web site u web pregledniku:

**Upravljanje sadržajem****Vježba 2**

Vježba se nalazi u pami cms2_kod.

Zadatak vježbe 2

1. Koristite znanje iz vježbe Knjiga gostiju i dodajte kôd koji će dovesti na web stranicu polje 'ime' svih redaka iz tablice 'podrucje'.

Rad

1. U mapu biblioteke kopirajte paket PEAR iz ranijih vježbi.
2. Za vježbu iz SQL-a, u MySQL naredbodavnom retku napravite SQL upit koji će izbaciti sve retke tablice 'podrucje' i stupce podrucje_id, ime.

Rješenje

1. Napravite mape:
 - a. podatkovni_objekti

- b. poslovni_objekti
- c. smarty_plugins

konfiguracija.inc.php

Dodajte ovaj kôd:

```
if ((substr(strtoupper(PHP_OS), 0, 3)) == "WIN")
    define("PATH_SEPARATOR", "\\");
else
    define("PATH_SEPARATOR", ":");
ini_set('include_path', SITE_ROOT . '/biblioteke/PEAR' .
PATH_SEPARATOR . ini_get('include_path'));
//podaci za logiranje u bazu
define("USE_PERSISTENT_CONNECTIONS", "true");
define("DB_SERVER", "localhost");
define("DB_USERNAME", "admin");
define("DB_PASSWORD", "admin");
define("DB_DATABASE", "prazna");
define("MYSQL_CONNECTION_STRING", "mysql://" . DB_USERNAME .
":" .
    DB_PASSWORD . "@" . DB_SERVER . "/" . DB_DATABASE);
?>
```

uspostavljanje_smarty.php

Dodajte ovaj kôd u metodu construct:

```
$this->plugins_dir[0] = SMARTY_DIR . 'plugins';
$this->plugins_dir[1] = SITE_ROOT . "/smarty_plugins";
```

vrh_aplikacije.php

Dodajte ovaj kôd:

```
require_once 'bazapodataka.php';

//Instanciranje globalnog objekta BazaPodataka
$gBazaPodataka = new BazaPodataka(MYSQL_CONNECTION_STRING);
```

bazapodataka.php

Kreirajte novi dokument u mapi include:

```
<?php
require_once 'DB.php';
class BazaPodataka
{
    public $bp;
    function __construct($connectionString)
    {
        $this->bp = DB::connect($connectionString,
                                USE_PERSISTENT_CONNECTIONS);
        $this->bp->setFetchMode(DB_FETCHMODE_ASSOC);
    }
    public function BpDisconnect()
    {
        $this->bp->disconnect();
    }
    public function BpQuery($queryString)
```

```

    {
        $result = $this->bp->query($queryString);
        return $result;
    }
    public function BpGetAll($queryString)
    {
        $result = $this->bp->getAll($queryString);
        return $result;
    }
    public function BpGetRow($queryString)
    {
        $result = $this->bp->getRow($queryString);
        return $result;
    }
    public function BpGetOne($queryString)
    {
        $result = $this->bp->getOne($queryString);
        return $result;
    }
}
?>

```

dno_aplikacije.php

```

<?php
$GLOBALS['gBazaPodataka']->BpDisconnect();
?>

```

index.tpl

Dodajte:

```

<body>
{include file="zaglavlje.tpl"}
{include file="podrucje.tpl"}
</body>

```

2. U mapi podatkovni objekti kreirajte ovaj dokument:

podatkovni_objekt.php

```

<?php
class PodatkovniObjekt
{
    function __construct()
    {
        $this->bPodataka = $GLOBALS['gBazaPodataka'];
    }
    public function GetPodrucje()
    {
        $query_string = "SELECT podrucje_id, ime FROM podrucje";
        $result = $this->bPodataka->BpGetAll($query_string);
        return $result;
    }
}
?>

```

3. U mapi poslovni objekti napravite ovaj dokument:

poslovni_objekt.php

```

<?php
require_once SITE_ROOT .
'/podatkovni_objekti/podatkovni_objekt.php';
class PoslovniObjekt
{
    private $mPodatkovniObjekt;
    function __construct()
    {
        $this->mPodatkovniObjekt = new PodatkovniObjekt();
    }
    public function GetPodrucje()
    {
        $result = $this->mPodatkovniObjekt->GetPodrucje();
        return $result;
    }
}
?>

```

4. Dodajte u:

index.php

```

<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>

```

function.load_podrucje.php

Ovaj novi dokument napravite u mapi smarty_plugins.

```

<?php
function smarty_function_load_podrucje($params, $smarty)
{
    $podrucje = new Podrucje();
    $podrucje->init();
    $smarty->assign($params['assign'], $podrucje);
}
class Podrucje
{
    public $mPodrucje;
    private $mPoslovniObjekt;
    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();
    }
    function init()
    {
        $this->mPodrucje = $this->mPoslovniObjekt->GetPodrucje();
    }
}
?>

```

podrucje.tpl

Ovaj novi dokument napravite u mapi templates:

```
{* podrucje.tpl *}
{load_podrucje assign="podrucje"}
{section name=i loop=$podrucje->mPodrucje}
    {$podrucje->mPodrucje[i].ime}
{/section}
```

Objašnjenje

SQL upit:

```
SELECT podrucje_id, ime FROM podrucje
```

Taj upit ugrađujemo u podatkovni sloj:

podatkovni_objekt.php

```
<?php
class PodatkovniObjekt
{
    function __construct()
    {
        $this->bPodataka = $GLOBALS['gBazaPodataka'];
    }
    public function GetPodrucje()
    {
        $query_string = "SELECT podrucje_id, ime FROM podrucje";
        $result = $this->bPodataka->BpGetAll($query_string);
        return $result;
    }
}
?>
```

Podatkovni sloj je uvijek pozivan SAMO od poslovnog sloja. Zato u poslovni sloj kodiramo istoimenu funkciju čiji je jedini zadatak POZVATI funkciju u podatkovnom sloju:

poslovni_objekt.php

```
<?php
require_once SITE_ROOT .
'/podatkovni_objekti/podatkovni_objekt.php';
class PoslovniObjekt
{
    private $mPodatkovniObjekt;
    function __construct()
    {
        $this->mPodatkovniObjekt = new PodatkovniObjekt();
    }
    public function GetPodrucje()
    {
        $result = $this->mPodatkovniObjekt->GetPodrucje();
        return $result;
    }
}
```

```
}
?>
```

Tko poziva poslovni sloj i funkciju GetPodrucje()? Poziva prezentacijski sloj. Tko čini prezentacijski sloj? Prezentacijski sloj UVIJEK čine dvije datoteka: jedna formata TPL i druga formata PHP. Naziv ove druge uvijek kreiramo ovako:
function.load_ime_datoteke_tip_TPL.

5. U mapi smarty_plugins napravite ovaj dokument:

function.load_podrucje.php

```
<?php
```

U ovom dokumentu je početak uvijek ovakav. Funkcija se uvijek zove **smarty_function_load_ime_datoteke_tipa_TPL** (\$params, \$smarty):

```
function smarty_function_load_podrucje($params, $smarty)
{
```

Kako programer razmišlja kad kreira ovu funkciju i njene varijable? Ovako. Ove Smarty funkcije imaju strogo određen redosljed instrukcija, pa tu programer nije kreativan nego rutiner. Ta rutina se sastoji u našem konkretnom slučaju u sljedećem:

instanciraj objekt Podrucje() iako klasa tek sljedi. To je tako u ovom slučaju gdje Smarty kreira predložak XHTML stranice.

```
$podrucje = new Podrucje();
```

pozovi metodu init(), koja pripada klasi Podrucje:

```
$podrucje->init();
```

varijabli \$smarty dodijeli varijablu \$područje na sljedeći način. Varijabla \$područje će imati podatke u obliku asocijativnog arraya:

```
array(
array('podrucje_id'=>'1', 'ime'=>'Hrvatski'),
array('podrucje_id'=>'2', 'ime'=>'Matematika'),
array('podrucje_id'=>'3', 'ime'=>'Fizika'),
array('podrucje_id'=>'4', 'ime'=>'Kemija'),
array('podrucje_id'=>'5', 'ime'=>'Zemljopis'),
array('podrucje_id'=>'6', 'ime'=>'Informatika'),
)
```

To će joj vratiti poslovni objekt, a on će to dobiti od podatkovnog objekta.

Podatkovni objekt je to uzeo iz baze podataka pomoću upita:

```
SELECT podrucje_id, ime FROM podrucje
```

Parametri \$params['assign'], \$podrucje znače da će asocijativni array biti razdjeljen u pod-arraye i tako predan varijabli \$smarty. U konkretnom slučaju ti pod-arrayi su:

```
array(                                array(
('podrucje_id'=>'1'),                ('ime'=>'Hrvatski'),
('podrucje_id'=>'2'),                ('ime'=>'Matematika'),
('podrucje_id'=>'3'),                ('ime'=>'Fizika'),
('podrucje_id'=>'4'),                ('ime'=>'Kemija'),
('podrucje_id'=>'5'),                ('ime'=>'Zemljopis'),
('podrucje_id'=>'6')                 ('ime'=>'Informatika')
```

```
)
)
```

Dakle, asocijativni array će preći u varijablu \$smarty kao grupa arraya:

```
$smarty->assign($params['assign'], $podrucje);
}
```

I evo tek sad stiže i klasa Podrucje koju smo već konkretizirali u objektu Podrucje i pohranili u varijabli \$podrucje:

```
class Podrucje
{
```

Ovo do sad je za programera rutina jer su to odredili autori Smartya. Sad mora programer misliti svojom glavom. Mora biti kreativan. Poslušajmo najprije autore Smartya, što se očekuje od ove klase i svih klasa njoj sličnih u ovoj aplikaciji.

Ova klasa mora pitati i dobiti od baze podatke u obliku asocijativnog arraya. I taj array razbiti u obične arraye i predati smarty predlošku, tj. datoteci tipa TPL.

- Dakle, klasi treba jedan javni član koji će pohraniti taj asocijativni array. Javan zato da bude dohvatljiv izvan svog objekta, konkretno dohvatljiv objektu smarty() koji će s arrayima kreirati predložak .TPL.

```
public $mPodrucje;
```

- Klasi treba i jedan privatni član koji će pohraniti instancu PoslovnogObjekta. Privatni član će živjeti samo koliko je potrebno da se procesuiru klasa:

```
private $mPoslovniObjekt;
```

Napomena: članovima smo ispred imena dodali malo slovo 'm' što potječe od riječi 'modul'. Klasu možemo smatrati i zatvorenom crnom kutijom ili modulom. Naime, od programera se samo traži da njihovi programi (klase) obave određene poslove. To je javno i to je ono što se prodaje: ova klasa radi to i to, ova radi to i to. A koje smo mi instrukcije ispisali u klasi to kupca ne zanima. Varijable ili članove ili svojstva klase označavamo malim slovom m ispred imena. Druga je sad stvar što programer određuje koliko će koji od tih članova živjeti s deklaracijama public, private, restricted.

Klasa Podrucje ima konstruktor. To je metoda koja će se izvršiti automatski kad se instancira objekt Podrucje:

```
function __construct()
{
```

Pseudo objekt (\$this = objekt Podrucje) poziva svog privatnog člana mPoslovniObjekt i dodijeljuje mu instancu objekta PoslovniObjekt.

```
$this->mPoslovniObjekt = new PoslovniObjekt();
}
```

Objekt Podrucje ima i metodu init(), koja radi sljedeće:

```
function init()
{
```

Pseudo objekt \$this poziva svog javnog člana mPodrucje s lijeve strane znaka jednakosti. S desne strane poziva privatnog člana mPoslovniObjekt, u kojem je

pohranjena instanca objekta PoslovniObjekt i poziva metodu GetPodrucje() tog PoslovnogObjekta. Iza ove jedne jedine instrukcije odigralo se sljedeće:

- Poslovni objekt je pozvao svoju funkciju GetPodrucje().
- Ta funkcija je pozvala PodatkovniObjekt i njegovu funkciju GetPodrucje().
- Bazi je upućen upit SELECT podrucje_id, ime FROM podrucje.
- Baza je vratila asocijativni array:

```
array(
array('podrucje_id'=>'1', 'ime'=>'Hrvatski'),
array('podrucje_id'=>'2', 'ime'=>'Matematika'),
array('podrucje_id'=>'3', 'ime'=>'Fizika'),
array('podrucje_id'=>'4', 'ime'=>'Kemija'),
array('podrucje_id'=>'5', 'ime'=>'Zemljopis'),
array('podrucje_id'=>'6', 'ime'=>'Informatika'),
)
```

- Podatkovna funkcija vraća taj array poslovnoj funkciji, a ova ga vraća metodi init() u konstruktoru objekta Podrucje.
- Asocijativni array se pohranjuje u javni član mPodrucje:

```
$this->mPodrucje = $this->mPoslovniObjekt->GetPodrucje();
}
}
```

Objekt Podrucje je predao klasi smarty() asocijativni array:

```
array(
array('podrucje_id'=>'1', 'ime'=>'Hrvatski'),
array('podrucje_id'=>'2', 'ime'=>'Matematika'),
array('podrucje_id'=>'3', 'ime'=>'Fizika'),
array('podrucje_id'=>'4', 'ime'=>'Kemija'),
array('podrucje_id'=>'5', 'ime'=>'Zemljopis'),
array('podrucje_id'=>'6', 'ime'=>'Informatika'),
)

?>
```

I sve se to događalo iza instrukcija koje su istaknute:

```
function smarty_function_load_podrucje($params, $smarty)
{
    $podrucje = new Podrucje();
    $podrucje->init();
    $smarty->assign($params['assign'], $podrucje);
}
```

Tad je ova instrukcija:

```
function smarty_function_load_podrucje($params, $smarty)
{
    $podrucje = new Podrucje();
    $podrucje->init();
    $smarty->assign($params['assign'], $podrucje);
}
```

dodijelila posebne arraye varijabli \$smarty

```
array(
('podrucje_id'=>'1'),
array(
('ime'=>'Hrvatski'),
```

```
( 'podrucje_id'=>'2'),      ('ime'=>'Matematika'),
('podrucje_id'=>'3'),      ('ime'=>'Fizika'),
('podrucje_id'=>'4'),      ('ime'=>'Kemija'),
('podrucje_id'=>'5'),      ('ime'=>'Zemljopis'),
('podrucje_id'=>'6')      ('ime'=>'Informatika')
)
```

Te Smarty varijable su iskorištene za izradu Smarty predloška web stranice, o čemu čitajte u nastavku.

6. U mapi templates napravite ovaj dokument:

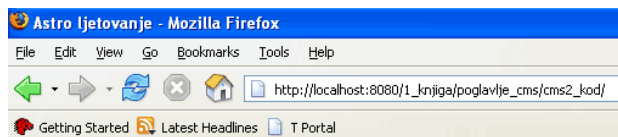
podrucje.tpl

```
{* podrucje.tpl *}
{load_podrucje assign="podrucje"}
{section name=i loop=$podrucje->mPodrucje}
    {$podrucje->mPodrucje[i].ime}
{/section}
```

Javni član `mPodrucje` prethodnog dokumenta `function.load_podrucje.php` pojavljuje se u ovom predlošku web stranice. Pomoću Smarty petlje koja se događa između tagova `SECTION` i `/SECTION`, iz člana `mPodrucje` se izvlače članovi samo arraya `'ime'`:

Hrvatski Matematika Fizika Kemija Zemljopis Informatika

7. Otvorite web site u web pregledniku:



Upravljanje sadržajem

Hrvatski Matematika Fizika Kemija Zemljopis Informatika

Područja su se upisala.

U sljedećoj vježbi od tih područja napravimo linkove.

Vježba 3

Vježba se nalazi u mapi `cms2_kod`.

Zadatak vježbe 3

1. Napraviti linkove od područja koja imamo na web stranici. To će biti glavna vodoravna navigacijska linija.

Rad

Pošto se radi samo o kozmetičkim zahvatima na prezentacijskom sloju, promjene će se i dogoditi samo u tom sloju.

function.load_podrucje.php

```
<?php
function smarty_function_load_podrucje($params, $smarty)
```

```

{
    $podrucje = new Podrucje();
    $podrucje->init();
    $smarty->assign($params['assign'], $podrucje);
}
class Podrucje
{
    public $mPodrucje;
    private $mPoslovniObjekt;
    public $mOdabranoPodrucje;

    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();

        if (isset($_GET['PodrucjeID']))
            $this->mOdabranoPodrucje = (int)$_GET['PodrucjeID'];
        else
            $this->mOdabranoPodrucje = -1;
    }
    function init()
    {
        $this->mPodrucje = $this->mPoslovniObjekt->GetPodrucje();
        for ($i = 0; $i < count($this->mPodrucje); $i++)
            $this->mPodrucje[$i]['ovo_je_link'] =
"index.php?PodrucjeID=" .
            $this->mPodrucje[$i]['podrucje_id'];
    }
}
?>

```

podrucje.tpl

```

{* podrucje.tpl *}
{load_podrucje assign="podrucje"}
{section name=i loop=$podrucje->mPodrucje}
    {if ($podrucje->mOdabranoPodrucje ==
        $podrucje->mPodrucje[i].podrucje_id)}
        {assign var=class_podrucje
value="PodrucjeOdabrano"}
    {else}
        {assign var=class_podrucje
value="PodrucjeNeodabrano"}
    {/if}
    <a class="{ $class_podrucje}"
        href="{ $podrucje->mPodrucje[i].ovo_je_link}">
        { $podrucje->mPodrucje[i].ime}
    </a> |
{/section}

```

stil.css

```

body
{
    font-family: verdana, arial, helvetica, sans-serif;
    font-size: small;
}

```

```

a.PodrucjeNeodabrano
{
    font-weight: bold;
    color: #5f9ea0;
    line-height: 25px;
    padding-left: 5px;
    text-decoration: none;
}
a.PodrucjeNeodabrano:hover
{
    padding-left: 5px;
    color: #000000;
    text-decoration: underline;
}
a.PodrucjeOdabrano
{
    font-weight: bold;
    color: #000000;
    line-height: 25px;
    padding-left: 5px;
    text-decoration: none;
}

```

1. Pogledajte sad web site u web pregledniku i vidjet ćete linkove.

Odaberite redom izbornike i vidjet ćete ove adrese linkova:

`http://localhost:8080/1_knjiga/poglavlje_cms/cms3_kod/index.php?PodrucjeID=1`

`http://localhost:8080/1_knjiga/poglavlje_cms/cms3_kod/index.php?PodrucjeID=2`

`http://localhost:8080/1_knjiga/poglavlje_cms/cms3_kod/index.php?PodrucjeID=3`

`http://localhost:8080/1_knjiga/poglavlje_cms/cms3_kod/index.php?PodrucjeID=4`

itd.

Objašnjenje

Najzanimljiviji dio kôda je onaj u kojem se formira link, u dokumentu

`function.load_podrucje.php`:

```

function init()
{
    $this->mPodrucje = $this->mPoslovniObjekt->GetPodrucje();
    for ($i = 0; $i < count($this->mPodrucje); $i++)
        $this->mPodrucje[$i]['ovo_je_link'] =
"index.php?PodrucjeID=" .
    $this->mPodrucje[$i]['podrucje_id'];
}

```


Pomoću petlje pronalazimo na koje smo područje kliknuli ('ovo_je_link') i onda pišemo konstantni dio linka: **index.php?PodrucjeID=**

Tom konstantnom dijelu dodajemo broj podrucje_id na koje smo kliknuli.

Vježba 4

Vježba se nalazi u mapi cms3_kod.

Zadatak ove vježbe je sljedeći:

1. Kako se formiraju adrese linkova?

Kreiranje URL

Cijeli CMS je jedna stranica: index.php. Sadržaj se mijenja ovisno o url upitu:

URL upit	Sadržaj stranice
http://localhost/cms_komplet/index.php	Vidjet će se skraćeni tekstovi onih jedinica koje imaju uključenu opciju Promidžba na HomePage
http://localhost/cms_komplet/index.php?PodrucjeID=1	Odabrali smo područje čije je podrucje_id = 1. Vidjet će se sve cjeline tog područja i svi skraćeni tekstovi jedinica koje imaju uključenu opciju Promidžba na razini područja.
http://localhost/cms_komplet/index.php?PodrucjeID=1&CjelinaID=1	Odabrali smo cjelinu čiji je podatak cjelina_id=1. Sadržaj stranice su SVE jedinice koje pripadaju toj cjelini. Jedinice su prikazane četiri po po četiri. Tu konstantu određujemo na stranici konfiguracija.inc.php: <code>define("JEDINICA_PO_STRANICI",4);</code>
http://localhost/cms_komplet/index.php?PodrucjeID=1&CjelinaID=1&JedinicaID=1	Odabrali smo jedinicu koja ima polje jedinica_id = 1. Ovo je hijerarhijski posljednji sadržaj našeg CMS. Ovdje je detaljno sve o jedinici.

Pogledajmo kôd u kojem je kreiran URL string:

index.php

```
<?php
require_once 'include/vrh_aplikacije.php';
```

URL string na poslužitelju hvata objekt GET i stvara onoliko varijabli koliko ima znakova jednakosti. Lijeva strana znaka jednakosti je ime varijable u GET objektu.

Na primjer, URL string:

```
http://localhost
/cms_komplet/index.php?PodrucjeID=1&CjelinaID=1&JedinicaID=1
```

će na poslužitelju kreirati ove varijable:

`$_GET['PodrucjeID'] = 1, $_GET['CjelinaID'] = 1, $_GET['JedinicaID'] = 1.`

Naš kôd koristi strukturu IF za izbacivanje odgovarajućeg SADRŽAJA. Ako u URL stringu postoje sve tri varijable, kôd će izbaciti detalje jedinice. Ako postoje samo dvije varijable, izbacit će sve cjeline i jedinice koje se promoviraju na razini područja – skraćene tekstove, i ako postoji samo jedna varijabla izbacit će sva područja i jedinice koje se promoviraju na razini Home Page.

Ako u stringu URL ne postoji JedinicaID, to znači da nismo na stranici detalja, a to onda znači da takvu adresu TREBA pohraniti u sesiju. Sve url osim stranice detalja se pohranjuju u sesiju, tako da se sa stranice detalja možemo uvijek vratiti na prethodnu adresu. Jer, mi do jedinice detalja možemo doći odmah s promidžbe na razini HomePage, možemo s promidžbe na razini područja, možemo preko tražilice i možemo doći postepeno preko područja i cjelina.

Dakle, ako nema varijable JedinicaID:

```
if (!isset($_GET['JedinicaID']))
```

Kreiraj sesiju 'LinkStranice'. Zahvaljujući ovoj sesiji, funkcionirat će link Povratak na prethodnu stranicu. Nije u vitičastim zagradama jer kad je samo jedna instrukcija u strukturi IF, ne treba stavljati vitičaste zagrade:

```
$_SESSION['LinkStranice'] = "http://" .
$_SERVER['SERVER_NAME'] .
":" . $_SERVER['SERVER_PORT'] . $_SERVER['REQUEST_URI'];
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
$home = new HomePage();
```

U ovom kôdu pada odluka što će biti prikazano na našoj jedinjoj web stranici index.php:

```
$stoCeBitiNaHomePage = "home_page.tpl";
$cjelinaDioStranice = "blank.tpl";
if (isset($_GET['PodrucjeID']))
{
    $stoCeBitiNaHomePage = "podrucje_opis.tpl";
    $cjelinaDioStranice = "cjelina.tpl";
}
if (isset($_GET['Search']))
    $stoCeBitiNaHomePage = "trazilica_rezultat.tpl";
if (isset($_GET['JedinicaID']))
    $stoCeBitiNaHomePage = "jedinica.tpl";
$home->assign("stoCeBitiNaHomePage", $stoCeBitiNaHomePage);
$home->assign("cjelinaDioStranice", $cjelinaDioStranice);
$home->display('index.tpl');
require_once 'include/dno_aplikacije.php';
?>
```

Linkovi

Linkovi su sastavni dio prezentacijskog sloja, web stranice. Zato linkove kreira Smarty, jer on kreira cijelu web stranicu. Prezentacijski sloj kreiraju dvije datoteke. Jedna je PHP a druga je TPL.

U slučaju 'područja', to su datoteke: **podrucje.tpl** i **function.load_podrucje.php**. Pogledajmo kako ove dvije datoteke grade linkove i kako ih formatiraju.

Uzeto iz mape cms_komplet.

function.load_podrucje.php

```
?php
function smarty_function_load_podrucje($params, $smarty)
{
    $podrucje = new Podrucje();
    $podrucje->init();
    $smarty->assign($params['assign'], $podrucje);
}
class Podrucje
{
    public $mPodrucje;
    private $mPoslovniObjekt;
```

U odnosu na prethodnu vježbu, sad nam treba još jedan javni član. To je član `$mOdabranoPodrucje` i on će smarty predlošku prenijeti `podrucje_id` od onog područja na koje mi kliknemo. Naime, kad napravimo linkove, svaki put kad kliknemo na neko područje, u polju Address web preglednika vidjet ćemo `index.php?PodrucjeID=1`, ili `index.php?PodrucjeID=6`, ovisno na koje smo područje kliknuli. Brojevi 1 do 6 su `podrucje_id` iz tablice 'podrucje'. `PodrucjeID` smo mi dodali adresi i to će biti varijabla `$_GET[PodrucjeID]`. PHP kôd će onda lako znati na koje smo područje kliknuli i to će pohraniti u varijablu `$mOdabranoPodrucje`.

```
    public $mOdabranoPodrucje;

    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();
```

Evo hvatanja vrijednosti varijable `$_GET['PodrucjeID']`. Ako postoji (isset) varijabla `$_GET['PodrucjeID']`, onda će funkcija `isset` izbaciti 'true'.

```
        if (isset($_GET['PodrucjeID']))
```

Ako je `isset()` izbacila 'true', dogodit će se ova instrukcija. Pomoću (int) će se osigurati da vrijednost varijable `PodrucjeID`, na primjer '1', stvarno bude integer 1.

```
            $this->mOdabranoPodrucje = (int)$_GET['PodrucjeID'];
        else
```

U sprotnom niti jedno područje neće biti odabrano, na niti jedno nismo kliknuli, što se definira s vrijednošću -1. `mOdabranoPodrucje` može imati vrijednost 0 kad je svijet arraya u pitanju, ali vrijednost -1 nikad. U našem slučaju, kad u tablici počinje brojenje od 1, nikad neće biti ni vrijednost 0, ali je uobičajeno staviti -1, pa smo mirni.

```
            $this->mOdabranoPodrucje = -1;
        }
```

U ovoj točki našeg programa javni član `mOdabranoPodrucje` čuva vrijednost `PodrucjeID`, na primjer integer 1.

```
function init()
{
```

Ova instrukcija je objašnjena u prošloj vježbi. Ona nam donosi asocijativni array.
`$this->mPodrucje = $this->mPoslovniObjekt->GetPodrucje();`

Ova petlja je novost. Ova petlja kreira još jedan asocijativni array uz pomoć već postojećeg arraya. Kako? Ovako:

Prolazi se kroz asocijativni array `mPodrucje` koji sadrži polja `podrucje_id` i `ime`. Redak po redak, od prvog do zadnjeg.

```
for ($i = 0; $i < count($this->mPodrucje); $i++)
```

Postojećem asocijativnom arrayu pohranjenom u varijabli `mPodrucje`:

```
array(
array('podrucje_id'=>'1', 'ime'=>'Hrvatski'),
array('podrucje_id'=>'2', 'ime'=>'Matematika'),
array('podrucje_id'=>'3', 'ime'=>'Fizika'),
array('podrucje_id'=>'4', 'ime'=>'Kemija'),
array('podrucje_id'=>'5', 'ime'=>'Zemljopis'),
array('podrucje_id'=>'6', 'ime'=>'Informatika'),
)
```

dodajemo još jedan pod-array tako što smo postojećoj varijabli pripisali po želji ključ, na primjer `'ovo_je_link'`:

```
$this->mPodrucje[$i]['ovo_je_link'] = "index.php?PodrucjeID=" .
$this->mPodrucje[$i]['podrucje_id'];
```

U ovoj točki našeg kôda imamo asocijativni array u varijabli `mPodrucje` koji se sastoji od TRI pod-arraya: `podrucje_id`, `ime`, `ovo_je_link`:

```
array(
array('podrucje_id'=>'1', 'ime'=>'Hrvatski', 'ovo_je_link'=>index.php?PodrucjeID=>'1'),
array('podrucje_id'=>'2', 'ime'=>'Matematika', 'ovo_je_link'=>index.php?PodrucjeID=>'2'),
array('podrucje_id'=>'3', 'ime'=>'Fizika', 'ovo_je_link'=>index.php?PodrucjeID=>'3'),
array('podrucje_id'=>'4', 'ime'=>'Kemija', 'ovo_je_link'=>index.php?PodrucjeID=>'4'),
array('podrucje_id'=>'5', 'ime'=>'Zemljopis', 'ovo_je_link'=>index.php?PodrucjeID=>'5'),
array('podrucje_id'=>'6', 'ime'=>'Informatika', 'ovo_je_link'=>index.php?PodrucjeID=>'6'),
)
```

```
    }
}
?>
```

Sad će smarty dobiti tri arraya za svoje predloške. Array `'ime'` će iskoristiti za ispis imena područja u vodoravnom retku. Array `'ovo_je_link'` će iskoristiti za kreiranje linkova u vodoravnom nizu.

Varijablu `mOdabranoPodrucje` će koristiti za dodijelu CSS klase

`PodrucjeOdabrano` jednom od područja, onom na koji smo kliknuli, a ostalima će dodijeliti CSS klasu `PodrucjeNeodabrano`.

Pogledajmo kako to smarty radi u predlošku `podrucje.tpl`:

podrucje.tpl

```
{* podrucje.tpl *}
{load_podrucje assign="podrucje"}
```

Petlja kroz asocijativni array `mPodrucje` koji sadrži tri pod-arraya.

```
{section name=i loop=$podrucje->mPodrucje}
```

Ako je broj `mPodrucje[i].podrucje_id` jednako onom u varijabli `mOdabranoPodrucje`, varijabla `class_podrucje` će imati vrijednost `PodrucjeOdabrano`.

```
{if ($podrucje->mOdabranoPodrucje ==
    $podrucje->mPodrucje[i].podrucje_id)}
    {assign var=class_podrucje
value="PodrucjeOdabrano"}
    {else}
```

U suprotnom, varijabla `class_podrucje` će imati vrijednost `PodrucjeNeodabrano`:

```
    {assign var=class_podrucje
value="PodrucjeNeodabrano"}
    {/if}
```

Još samo da kreiramo linkove za sva područja na našoj web stranici. Linkovi se kreiraju s tagovima `<a> i `. Argumenti taga `<a` mogu biti `class='PodrucjeOdabrano'` ili `class='PodrucjeNeodabrano'`, te `href=index.php?PodrucjeID=1'Hrvatski`

`'index.php?PodrucjeID=2'>Matematika`

`'index.php?PodrucjeID=3'>Fizika`

`'index.php?PodrucjeID=4'>Kemija`

`'index.php?PodrucjeID=5'>Zemljopis`

`'index.php?PodrucjeID=6'>Informatika`

```
    <a class="{ $class_podrucje}"
    href="{ $podrucje->mPodrucje[i].ovo_je_link}">
    { $podrucje->mPodrucje[i].ime}
    </a> |
{/section}
```

Vježba 5

Ovaj kôd se nalazi u mapi `cms_komplet`.

Zadatak:

1. Kako prikazujemo cjeline, točnije, polje 'ime' iz tablice 'cjelina', za određeno područje, točnije `podrucje_id`?
2. Napišimo SQL upit u MySQL naredbodavnom retku.

SQL upit glasi:

```
SELECT cjelina_id, ime
      FROM cjelina
     WHERE podrucje_id = 1;
```

Rad

Ovdje se sve događa kao što je opisano i u prethodnim vježbama. Za određeno područje, kreira se okomiti izbornik sastavljen od linkova imena cjelina.

cjelina.tpl

```
{* cjelina.tpl *}
{load_cjelina assign="cjelina"}
    {section name=i loop=$cjelina->mCjelina}
        {assign var=class_cjelina value="CjelinaNeodabrana"}
```

```

        {if ($cjelina->mCjelinaOdabrana ==
        $cjelina->mCjelina[i].cjelina_id)}
        {assign var=class_cjelina value="CjelinaOdabrana"}
        {/if}
        <a class="{ $class_cjelina}"
        href="{ $cjelina-
        >mCjelina[i].ovo_je_link|escape:"html"}">
        { $cjelina->mCjelina[i].ime}
        </a>
    <br />
        {/section}
    {* kraj cjelina.tpl *}

```

function.load_cjelina.php

```

<?php
function smarty_function_load_cjelina($params, $smarty)
{
    $cjelina = new Cjelina();
    $cjelina->init();
    $smarty->assign($params['assign'], $cjelina);
}
class Cjelina
{
    /* javne varijable za predloške (template) smarty */
    public $mCjelinaOdabrana = 0;
    public $mPodrucjeOdabrano = 0;
    public $mCjelina;
    /* privatni članovi */
    private $mPoslovniObjekt;
    /* constructor */
    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();
        if (isset($_GET['PodrucjeID']))
            $this->mPodrucjeOdabrano = (int)$_GET['PodrucjeID'];
        if (isset($_GET['CjelinaID']))
            $this->mCjelinaOdabrana = (int)$_GET['CjelinaID'];
    }
    /* init */
    function init()
    {
        $this->mCjelina =
        $this->mPoslovniObjekt->GetCjelineUPodrucju($this-
        >mPodrucjeOdabrano);
        //izgradnja linkova za stranice `cjelina`
        for ($i = 0; $i < count($this->mCjelina); $i++)
            $this->mCjelina[$i]['ovo_je_link'] =
                "index.php?PodrucjeID=" .
                $this->mPodrucjeOdabrano . "&CjelinaID=" .
                $this->mCjelina[$i]['cjelina_id'];
    }
} //kraj klase
?>

```

Objašnjenje

cjelina.tpl

```
{* cjelina.tpl *}
```

Učitavanje vrijednosti varijabli iz dokumenta function.load_cjelina.php i dodijeljivanje istih Smartyevoj varijabli cjelina:

```
{load_cjelina assign="cjelina"}
```

Počinja petlja pomoću koje ćemo izvući posebne arraye iz asocijativnog arraya mCjelina. mCjelina je varijabla koja je ovdje dospijela iz function.load_cjelina.php, i u sebi nosi sve retke iz tablice 'cjelina':

```
{section name=i loop=$cjelina->mCjelina}
```

Kreiramo varijablu class_cjelina koja će imati dvije vrijednosti CjelinaNeodabrana ili CjelinaOdabrana. U CSS listu smo kreirali class CjelinaNeodabrana i CjelinaOdabrana:

```
{assign var=class_cjelina value="CjelinaNeodabrana"}
```

Na koju smo cjelinu kliknuli, to je uhvaćeno u function.load_cjelina.php i preko varijable mCjelinaOdabrana prenijeto ovdje:

```
{if ($cjelina->mCjelinaOdabrana ==
$cjelina->mCjelina[i].cjelina_id)}
{assign var=class_cjelina value="CjelinaOdabrana"}
{/if}
```

Svaki naš klik pokreće cijeli proces, pa se tako ponovo procesuiru i stranica function.load_cjelina.php i hvata koju smo cjelinu odabrali. Ovdje se susrećemo i s escape:"html" što znači "u izvornom kôdu piši &, u url piši &".

Iz dokumenta function.load_cjelina.php je pristigao i pod-array **mCjelina[i].ovo_je_link** asocijativnog arraya mCjelina. Primjer: u sebi sadrži sva 4 linka za sve 4 cjeline za područje Hrvatski kojem je podrucje_id = 1.

```
<a class="{ $class_cjelina}"
href="{ $cjelina->mCjelina[i].ovo_je_link|escape:"html"}">
    { $cjelina->mCjelina[i].ime}
</a>
<br />
{/section}
{* kraj cjelina.tpl *}
```

Pogledajte izvorni kod stranice i vidjet ćete učinak ovog Smarty koda, na primjeru 4 linka:

```
<a class="CjelinaOdabrana"
href="index.php?PodrucjeID=1&CjelinaID=1">Književnost</a>
<br />
<a class="CjelinaNeodabrana"
href="index.php?PodrucjeID=1&CjelinaID=2">Pravopis</a>
<br />
<a class="CjelinaNeodabrana"
href="index.php?PodrucjeID=1&CjelinaID=3">Proza</a>
<br />
<a class="CjelinaNeodabrana"
```

```
href="index.php?PodrucjeID=1&CjelinaID=4">Lektira</a>
```

Kad kliknemo na područje Hrvatski i na cjelinu Književnost dobit ćemo na primjer ovaj url u address preglednika:

```
index.php?PodrucjeID=1&CjelinaID=1
```

Pogledajte izvorni kod stranice i vidjet ćete:

```
<a class="CjelinaOdabrana"
href="index.php?PodrucjeID=1&CjelinaID=1">Književnost</a>
```

Ovdje piše **&**, a ne piše **&**, kao što će pisati u url.

function.load_cjelina.php

```
<?php
```

Ovo je nama već jako dobro poznat početak dokumenta function.load_cjelina.php:

```
function smarty_function_load_cjelina($params, $smarty)
{
    $cjelina = new Cjelina();
    $cjelina->init();
    $smarty->assign($params['assign'], $cjelina);
}
class Cjelina
{
    /* javne varijable za predloške (template) smarty */
```

Evo članova koji će prenijeti smarty predlošku (.TPL) podatke o tome na koju smo cjelinu kliknuli, koje je područje odabrano, i asocijativni array iz baze plus dodani pod-array za linkove:

```
public $mCjelinaOdabrana = 0;
public $mPodrucjeOdabrano = 0;
public $mCjelina;
/* privatni članovi */
private $mPoslovniObjekt;
/* constructor */
function __construct()
{
```

I ovo već znamo na pamet. Nalazimo se u prezentacijskom sloju. Znamo na koju cjelinu je učinjeno klik. Pozivamo PoslovniObjekt:

```
$this->mPoslovniObjekt = new PoslovniObjekt();
```

Utvrđujemo brojeve područja i cjeline iz url:

index.php?PodrucjeID=6&CjelinaID=1 i osiguravamo da budu tip podataka integer pomoću (int):

```
if (isset($_GET['PodrucjeID']))
    $this->mPodrucjeOdabrano = (int)$_GET['PodrucjeID'];
if (isset($_GET['CjelinaID']))
    $this->mCjelinaOdabrana = (int)$_GET['CjelinaID'];
}
```

Smarty metoda init() sad odrađuje svoj rutinski posao:

```
/* init */
function init()
{
```


Javnom članu `mCjelina` dodijeljuje asocijativni array koji će se vratiti preko slojeva podatkovnog i poslovnog. Za to je zaslužna metoda `GetCjelineUPodrucju($this->mPodrucjeOdabrano)` u poslovnom i podatkovnom sloju.

Parametrom `$this->mPodrucjeOdabrano` mi smo toj metodi prenijeli vrijednost `podrucje_id`. SQL upit u podatkovnom sloju glasi ovako:

```
$this->mCjelina =
    $this->mPoslovniObjekt->GetCjelineUPodrucju($this-
        >mPodrucjeOdabrano);
```

SQL upit koristi `podrucje_id` za SELECT svih redaka u tablici CJELINA koji pripadaju tom području. SQL upit je u podatkovnom sloju i glasi ovako:

```
public function GetCjelineUPodrucju($podrucjeId)
{
    $query_string = "SELECT cjelina_id, ime
                     FROM cjelina
                     WHERE podrucje_id = $podrucjeId";
    $result = $this->bPodataka->BpGetAll($query_string);
    return $result;
}
```

I evo konačno mjesta gdje se događa sljedeće. Mi se nalazimo u klasi `Cjelina`. Toj klasi pripada i metoda `init()` u kojoj smo sad. Objekt `Cjelina` je instanciran davno, pogledajte na vrh ovog dokumenta. Javnom članu `mCjelina` je već pristigao asocijativni array koji se sastoji od dva arraya: `cjelina_id`, `ime` (to vidimo iz SQL upita SELECT). I što se sad događa? Pomoću petlje FOR prolazi se kroz asocijativni array `mCjelina`. Arrayi dobijaju indexe od 0 do 5, ako ima na primjer 6 `cjelina`. Array ima i podatak `count` koji je jednak broju članova arraya, znači 6. Zato broj prolaza petlje ide od `i=0` do `i< count($this->mCjelina) = 6`, znači od `i=0` do `i=5`, a to je 6 prolaza za svih 6 članova.

Linkovi se pohranjuju u posebni array. Imamo već asocijativni array `mCjelina` koji ima pod-arraye: `mCjelina[cjelina_id]`, `mCjelina[ime]`. Sad ćemo članu `mCjelina` dodati i treći pod-array, pod ključem '`ovo_je_link`', `mCjelina[ovo_je_link]`:

```
//izgradnja linkova za stranice `cjelina`
for ($i = 0; $i < count($this->mCjelina); $i++)
    $this->mCjelina[$i]['ovo_je_link'] =
        "index.php?PodrucjeID=" .
        $this->mPodrucjeOdabrano . "&CjelinaID=" .
        $this->mCjelina[$i]['cjelina_id'];
}
} //kraj classe
?>
```

Smarty predlošku `cjelina.tpl` bit će prenijet ovaj pod-array:

```
mCjelina[$i]['ovo_je_link']
```

Ta varijabla `mCjelina[$i]['ovo_je_link']` sadrži array ovih vrijednosti, ako postoje samo 4 cjeline:

```
index.php?PodrucjeID=6&CjelinaID=1
index.php?PodrucjeID=6&CjelinaID=2
```

index.php?PodrucjeID=6&CjelinaID=3

index.php?PodrucjeID=6&CjelinaID=4

Predložak **cjelina.tpl** će od arraya **mCjelina**[\$i]['ovo_je_link'] kreirati ovakve linkove:

```
<a class="CjelinaOdabrana"
href="index.php?PodrucjeID=1&CjelinaID=1">Književnost</a>
<br />
<a class="CjelinaNeodabrana"
href="index.php?PodrucjeID=1&CjelinaID=2">Pravopis</a>
<br />
<a class="CjelinaNeodabrana"
href="index.php?PodrucjeID=1&CjelinaID=3">Proza</a>
<br />
<a class="CjelinaNeodabrana"
href="index.php?PodrucjeID=1&CjelinaID=4">Lektira</a>
```

Vježba 6

Zadatak:

1. Kako prikazujemo jedinice?

function.load_jedinica.php

```
<?php
function smarty_function_load_jedinica($params, $smarty)
{
    $jedinica = new Jedinica();
    $jedinica->init();
    // assign template variable
    $smarty->assign($params['assign'], $jedinica);
}
class Jedinica
{
    // javne varijable koje će koristiti Smarty za izradu
    predloška
    public $mJedinica;
    public $mPageLink = "index.php";
    /* privatne varijable koje će živjeti koliko traje
    procesuiranje ove klase*/
    private $mPoslovniObjekt;
    private $mJedinicaId;

    function __construct()
    {
        // kreiranje objekta srednjeg sloja
        $this->mPoslovniObjekt = new PoslovniObjekt();
        /* inicijalizacija varijable, ili davanje vrijednosti
        varijabli. Kojih vrijednosti? Onih koje postoje u objektu GET
        */
        if (isset($_GET['JedinicaID']))
            $this->mJedinicaId = (int)$_GET['JedinicaID'];
    }
    // init
```

```
function init()
{
    // dobijanje podataka neke jedinice
    $this->mJedinica =
        $this->mPoslovniObjekt->GetJedinicaDetalji($this-
        >mJedinicaId);
```

Ako postoji varijabla LinkStranice u sesiji, pohranimo je u javni član mPageLink. Na smarty predlošku ćemo pomoću tog člana kreirati link Povratak.

```
        if (isset($_SESSION['LinkStranice']))
            $this->mPageLink = $_SESSION['LinkStranice'];
    }
} //kraj klase
?>
```

jedinica.tpl

```
{load_jedinica assign="jedinica"}
<h1>{$jedinica->mJedinica.ime}</h1>
<p>
<a class="link_povratak" href="{$jedinica-
>mPageLink}">Povratak</a>
</p>
<!--
        <img align="left" src='jedinica_slike/{$jedinica-
>mJedinica.slika_1}'
            height="150" border="0" alt="{$jedinica-
>mJedinica.slika_1}">

--->
        <div class="slika">
            <img align="left" src='jedinica_slike/{$jedinica-
>mJedinica.slika_2}'
                border="0" alt="{$jedinica->mJedinica.slika_1}"
            />
        </div>
<p>
<span class="JedinicaOpis">{$jedinica->mJedinica.opis}
<br />
<a class="link_povratak" href="{$jedinica-
>mPageLink}">Povratak</a></p>
```

Vježba 7

Zadatak

1. Kako smo prikazali promidžbene jedinice na razini HomePage?

home_page.tpl

```
{* home_page.tpl *}

<br />
    Ovo je primjer CMS-a! Nadamo se da će vam se svidjeti!
<br />
    <a href="admin.php">Administracija ovog sitea</a>
<br />
```

```
<br />
{include file="jedinica_kratko.tpl"}
```

jedinica_kratko.tpl

Ovaj Smarty predložak razmješta sadržaj koji dobije preko Smarty varijabli. Tu će se prikazati samo dio teksta iz opisa jedinica, što je određeno našim konstantama, i pojaviti će se samo 4 jedinice po stranici što opet mijenjamo u dokumentu u kojem su konstante. Zato se u ovom Smarty kôdu provjerava koliko ima ukupno stranica, kreiraju se linkovi Sljedeća, Prethodna itd.

```
{* jedinica kratko.tpl *}
{load_jedinica_kratko assign="lista_jedinica"}
<span class="ListaSearchTitle">{$lista_jedinica-
>mSearchResultsTitle}
</span>
{if $lista_jedinica->mrTotalPages > 1}
  <br />
  <span class="PrijelomTeksta">
    Stranica {$lista_jedinica->mPageNo} od ukupno
    {$lista_jedinica->mrTotalPages}:
    {if $lista_jedinica->mPreviousLink}
      <a href='{$lista_jedinica-
>mPreviousLink|escape:"html"}'>Prethodna</a>
    {else}
      Prethodna
    {/if}
    {if $lista_jedinica->mNextLink}
      <a href='{$lista_jedinica-
>mNextLink|escape:"html"}'>Sljedeća</a>
    {else}
      Sljedeća
    {/if}
  <br /><br />
</span>
{/if}
<table cellpadding="0" border="0">
  <tr>
    {section name=k loop=$lista_jedinica->mJedinica}
    {if $smarty.section.k.index !=0 &&
$smarty.section.k.index % 2 == 0}
  </tr>
  <tr>
    {/if}
    <td>
      <table cellpadding="0" align="left">
        <tr>
          <td align="right">
            <a href='{$lista_jedinica-
>mJedinica[k].ovo_je_link|escape:"html"}'>
              <img src='jedinica_slike/{$lista_jedinica-
>mJedinica[k].slika_1}'
                height="150" border="0" width="120" alt="slika
jedinice" />
            </a>
          <br /><br />
```

```

        </td>
        <td valign="top" width="200">
            <span class="JedinicaIme">
                <a href="{ $lista_jedinica-
>mJedinica[k].ovo_je_link|escape:"html"}"
                class="JedinicaIme">
                    { $lista_jedinica->mJedinica[k].ime}
                </a>
            </span>
            <br /><br />
            <span class="JedinicaOpis">
                { $lista_jedinica->mJedinica[k].opis}
            <br /><br />
            </span>
        </td>
    </tr>
</table>
</td>
{/section}
</tr>
</table>

```

function.load_jedinica_kratko.php

Ovaj php kôd snabdijeva Smarty predložak sa sadržajem. Taj sadržaj pristiže preko podatkovnog i poslovnog sloja. Sadržaj uvijek dolazi kao asocijativni array. Iz tog asocijativnog arraya, ovaj dokument gradi sastavne arraye i pohranjuje ih u članove objekta. Ti članovi objekta prenose te arraye do Smarty predloška, koji pomoću petlji vadi pojedine članove arraya i razmješta ih u XHTML strukturu.

Zaključak: Prezentacijski sloj grade dva Smarty dokumenta. Jedan je .tpl a drugi .php. U .php dokument UVIJEK pristiže odgovor na SQL UPIT u obliku ASOCIJATIVNOG ARRAYA. Zadatak .php dokumenta je dalje taj asocijativni array rastaviti u pod arraye, pohraniti u javne članove, i te javne članove predati .tpl dokumentu. TPL dokument iz tih OBIČNIH ARRAYA pomoću petlje FOR izvlači pojedine članove arraya i ugrađuje ih u XHTML strukturu web stranice.

```

<?php
function smarty_function_load_jedinica_kratko($params,
$smarty)
{
    $jedinica_kratko = new JedinicaKratko();
    $jedinica_kratko->init();
    // dodijeljivanje varijabli smartyjevu predlošku
    $smarty->assign($params['assign'], $jedinica_kratko);
}
class JedinicaKratko
{
    /* javne varijable koje će pročitati Smarty za kreiranje
    predloška */

```

Evo javnih članova koji će prenijeti sadržaj do Smarty predloška.

```

public $mJedinica;
public $mPageNo;

```

```

public $mrTotalPages;
public $mNextLink;
public $mPreviousLink;
public $mSearchResultsTitle;
public $mSearch = "";
public $mAllWords = "off";
/* privatne varijable */

```

Privatni članovi će živjeti samo dok se izvršava metoda construct. To su privatni članovi objekta i služe u metodi construct za stvaranje javnih članova koji ostaju na životu i nakon što se metoda construct izvrši.

```

private $mPoslovniObjekt;
private $mPodrucjeId;
private $mCjelinaId;
/* konstruktor */
function __construct()
{
    // kreiranje objekta poslovnog sloja

```

Svi upiti (url stringovi) se iniciraju u prezentacijskom sloju. Mi smo sad u prezentacijskom sloju. Klik na nešto i u address polju možemo sve vidjeti, cijeli URL string. Taj URL string hvata na serveru objekt GET i izvlači varijable. Svakoj kombinaciji varijabli GET, tj. za svaki mogući URL, kreirali smo i SQL UPITE u podatkovnom sloju. Do podatkovnog sloja se dolazi SAMO preko poslovnog. Zato prezentacijski sloj UVIJEK poziva POSLOVNI SLOJ. Nikad ne može stupiti u direktni kontakt s podatkovnim slojem. Pa tako i ovaj PHP dokument poziva poslovni sloj ovako:

```

$this->mPoslovniObjekt = new PoslovniObjekt();
// uzeti PodrucjeID iz url query stringa i pomoću (int)
osigurati da je to integer

```

A sad treba s mnogo IF struktura otkriti s kojih pozicija dolazi poziv za jedinicom_kratko. A poziv može doći s pozicija promidžbe na HomePage, promidžbe na razini područja, s redovne razine kad su sve jedinice prikazane (četiri po četiri ako ih je više od četiri). Poziv može doći i preko tražilice. Tražilica pretražuje tablicu 'jedinica' stupce ime i opis. Tražilica radi na dva načina:

- a. izbacuje sve jedinice_kratko u kojima zatiče bilo koju od riječi u tražilici, ili
- b. izbacuje sve jedinice_kratko u kojima zatiče SVE riječi u tražilici.

Kad smo odredili odakle dolazi poziv, onda se zna koja će se metoda pozvati u poslovnom sloju, jer će ta pozvati istoimenu metodu u podatkovnom sloju. Podatkovni sloj u toj metodi sadrži SQL string pomoću kojeg izvlači podatke iz baze, vraća podatke poslovnom sloju, a ovaj ih vraća prezentacijskom sloju.

Dakle, kliknuli smo na nešto i tako kreirali URL string. Taj url string na poslužitelju hvata objekt GET i formira odgovarajuće GET varijable. Ovisno o GET varijablama, pozvat će se odgovarajući SQL upit i dobiti odgovarajući sadržaj iz baze.

Svi IF i GET koji sljede orijentir su nam što sve možemo imati u našem url u ovoj CMS aplikaciji:

```

        if (isset($_GET['PodrucjeID']))
            $this->mPodrucjeId = (int)$_GET['PodrucjeID'];
        // uzeti CjelinaID iz url query stringa i pomoću (int)
osigurati da je to integer
        if (isset($_GET['CjelinaID']))
            $this->mCjelinaId = (int)$_GET['CjelinaID'];
        // uzeti PageNo iz url query stringa i pomoću (int)
osigurati da je to integer
        if (isset($_GET['PageNo']))
            $this->mPageNo = (int)$_GET['PageNo'];
        else
            $this->mPageNo = 1;
            // uzeti Search iz url query stringa
        if (isset($_GET['Search']))
            $this->mSearchString=$_GET['Search'];
        // uzeti vrijednost AllWords iz url query stringa
        if (isset($_GET['AllWords']))
            $this->mAllWords=$_GET['AllWords'];
    }
    /* init */
    function init()
    {
        // ako postoji string iz tražilice, preko poslovnog
sloja dođimo do rezultata
        if (isset($this->mSearchString))
        {
            $search_results = $this->mPoslovniObjekt->Search(
$this->mSearchString, $this->mAllWords,
$this->mPageNo, $this->mrTotalPages);
            // dobiti listu jedinica od objekta SearchResults
$this->mJedinica = & $search_results->mJedinica;
            // formirati listu naslova jedinica
            if (!empty($search_results->mSearchedWords))
                $this->mSearchResultsTitle =
                    "Jedinice koje sadrže <font color=\"red\">"
                    . ($this->mAllWords == "on" ? "sve navedene
riječi" :
                    "bilo koju od navedenih riječi") . "</font>"
                    . " <font color=\"red\">"
                    . $search_results->mSearchedWords .
"</font><br>";
                if (!empty($search_results->mIgnoredWords))
                    $this->mSearchResultsTitle .=
                        "Zanemarene riječi: <font color=\"red\">"
                        . $search_results->mIgnoredWords .
"</font><br/>";
                if (empty($search_results->mJedinica))
                    $this->mSearchResultsTitle .=
                        "Nema traženih riječi.<br/>";
                $this->mSearchResultsTitle .= "<br/>";
            }
        }
    }

```

```

        // kad kliknemo na link Cjelina, sve jedinice koje
        pripadaju toj cjelini
        // bit će učitane u javnu varijablu mJedinica, preko
        metode
        // GetJedinicaUCjelini u poslovnom sloju
        elseif (isset($this->mCjelinaId))
            $this->mJedinica = $this->mPoslovniObjekt-
>GetJedinicaUCjelini($this
            ->mCjelinaId, $this->mPageNo, $this->mrTotalPages);
        // kad kliknemo na link Podrucje, sve jedinice koje se
        promoviraju u tom Podrucju
        // bit će učitane u javnu varijablu mJedinica, preko
        poslovnog sloja i
        // metode GetJediniceNaPromidzbiNaRaziniPodrucja
        elseif (isset($this->mPodrucjeId))
            $this->mJedinica =
                $this->mPoslovniObjekt-
>GetJediniceNaPromidzbiNaRaziniPodrucja
                ($this->mPodrucjeId, $this->mPageNo, $this-
>mrTotalPages);
        // kad kliknemo na link za početnu stranicu, sve jedinice
        koje se promoviraju
        // na prvoj stranici
        // bit će učitane u javnu varijablu mJedinica, preko
        metode
        // GetPromidzbaJedinicaNaHomePage u poslovnom sloju
        else
            $this->mJedinica =
                $this->mPoslovniObjekt-
>GetPromidzbaJedinicaNaHomePage($this->mPageNo,

$this->mrTotalPages);
        // ako postoji više stranica, prikaži linkve sljedeća i
        prethodna
        if ($this->mrTotalPages > 1)
        {
            // učitaj query string
            $query_string = $_SERVER['QUERY_STRING'];
            // provjerimo da li postoji PageNo u query stringu
            $pos = stripos($query_string, "PageNo=");
            // ako PageNo ne postoji u query stringu
            // to znači da smo na prvoj stranici
            if ($pos == false)
            {
                $query_string .= "&PageNo=1";
                $pos = stripos($query_string, "PageNo=");
            }
            // pročitajmo tekući broj stranice iz query stringa
            $temp = substr($query_string, $pos);
            sscanf($temp, "PageNo=%d", $this->mPageNo);
            // kreirajmo link Sljedeća
            if ($this->mPageNo >= $this->mrTotalPages)
                $this->mNextLink = "";
            else
            {

```



```

        $new_query_string = str_replace("PageNo=" . $this-
>mPageNo,
                                "PageNo=" . ($this->mPageNo + 1),
        $query_string);
        $this->mNextLink = "index.php?". $new_query_string;
    }
    // kreirajmo link Prethodna
    if ($this->mPageNo == 1)
        $this->mPreviousLink = "";
    else
    {
        $new_query_string = str_replace("PageNo=" . $this-
>mPageNo,
                                "PageNo=" . ($this->mPageNo - 1),
        $query_string);
        $this->mPreviousLink =
        "index.php?". $new_query_string;
    }
}
// kreirajmo linkove za stranicu jedinica_detalji
$url = $_SESSION['LinkStranice'];
if (count($_GET) > 0)
    $url = $url . "&JedinicaID=";
else
    $url = $url . "?JedinicaID=";
for ($i = 0; $i < count($this->mJedinica); $i++)
    $this->mJedinica[$i]['ovo_je_link'] =
        $url . $this-
>mJedinica[$i]['jedinica_id'];
}
} //kraj klase
?>

```

Vježba 8

Kôd se nalazi u mapi cms_komplet.

Tražilica

U našem CMS ćemo pretraživati samo stupce 'ime' i 'opis' u tablici 'jedinica'.

Radi bržeg pretraživanja, kreirali smo index nad ta dva stupca i funkcionalnost FULLTEXT pomoću ovog SQL upita:

```
ALTER TABLE jedinica ADD FULLTEXT ImeOpisFTIndex(ime, opis);
```

Funkcionalnost FULLTEXT je puno bolja nego pretraživanje pomoću LIKE. Brža je i što je još važnije, FULLTEXT izbacuje relevantnu listu prema broju traženih riječi koje neki tekst može imati. Više riječi iz tražilice u tekstu, tekst će biti bliže vrhu liste.

MySQL omogućava pretraživanje po 'bilo kojoj riječi' i po 'svim riječima'. Obje opcije su na raspolaganju posjetitelju našeg CMS-a.

Napomena: Ako se traženi pojam pojavljuje u više od 50% svih redaka u tablici, moguće je da vaša tražilica ne izbaci niti jedan redak u jednoj od opcija AllWords

ili AnyWords.. Meni se to dogodilo. Ako se traženi pojam pojavi u manje od 3 retka tablice, također je moguće ne pojavljivanje niti jednog retka u relevantnoj listi u jednoj od opcija AllWords ili AnyWords.

Na osnovu naučenog u prethodnim vježbama, neće trebati puno riječi za objašnjenje kôda koji sudjeluje u tražilici.

Prezentacijski sloj, od kojeg sve počinje, čine smarty predlošci: trazilica_forma.tpl i njen par function.load_trazilica_forma.php, te trazilica_rezultat.tpl koja koristi postojeći par function.load_jedinica_kratko.php i jedinica_kratko.tpl kojeg mo već analizirali ranije.

trazilica_forma.tpl

```
{* trazilica_forma.tpl *}
{load_trazilica_forma assign="search_box"}
  <form class="forma_trazilica" action="index.php">
    <input class="trazilica" maxlength="100" id="Search"
name="Search"
    value="{ $search_box->mSearchString}" size="22"/>
    <input class="trazilica" type="submit"
value="Traži!"/><br />
    <input type="checkbox" id="AllWords" name="AllWords"
    {if $search_box->mAllWords == "on" } checked="checked"
  {/if}/>
    Sve riječi zastupljene
  </form>
{* kraj trazilica_forma.tpl *}
```

function.load_trazilica_forma.php

```
<?php
function smarty_function_load_trazilica_forma($params,
$smartyy)
{
    $search_box = new SearchBox();
    $smarty->assign($params['assign'], $search_box);
}
// classa koja upravlja s formom tražilice
class SearchBox
{
    /* javne varijable za smarty predložak */
    public $mSearchString = "";
    public $mAllWords = "off";
    /* constructor */
    function __construct()
    {
        $this->mPoslovniObjekt = new PoslovniObjekt();
        if (isset($_GET['Search']))
            $this->mSearchString = $_GET['Search'];
        if (isset($_GET['AllWords']))
            $this->mAllWords = $_GET['AllWords'];
    }
} //kraj classe
?>
```

trazilica_rezultat.tpl

```
{* trazilica_rezultat.tpl *}
Rezultat pretraživanja
<p>
    {include file="jedinica_kratko.tpl"}
</p>
```

poslovni_objekt.php

Jedino za tražilicu, poslovni objekt nije samo pozivatelj funkcija u podatkovnom objektu. Tražilica je i taj sloj zaposlila. Evo tog kôda u poslovnom objektu.

```
// pretraži bazu
```

Funkcija ima 4 parametra. To su ove varijable: \$searchString, \$allWords, \$pageNo, &\$rTotalPages. Prva tri parametra su kao i svi do sad koje smo koristili, a to je unos vrijednosti izvana, na engleskom 'by value'. Vrijednost izvana, iz jednog memorijskog mjesta, prelazi u funkciju u drugo memorijsko mjesto. Iz jedne varijable u drugu. Zato se može desiti da istovremeno jedna te ista vrijednost postoji na više mjesta u memoriji. Četvrti parametar ima ispred & što znači da se prenosi 'by reference'. To znači da se ne prenosi vrijednost iz jedne varijable u neku varijablu unutar funkcije, nego se prenosi SAMO memorijska adresa mjesta gdje se nalazi ta vrijednost.

Ta četiri parametra prenose sljedeće 4 vrijednosti: searchString su riječi koje tipkamo u formu tražilice, allWords je uključena/isključena opcija "traži SAMO tekstove koji sadrže SVE riječi u stringu", pageNo je broj tekuće stranice (kad ima više od 4 jedinice, svake 4 jedinice su jedna stranica) i rTotalPages je ukupan broj stranica (na primjer ako se na tražilicu vrati 16 jedinica, bit će ukupno 4 stranice).

```
public function Search($searchString, $allWords, $pageNo,
    &$rTotalPages)
{
    // kreiraj instancu objekta SearchResults
    $search_results = new SearchResults();
    // mogući znak koji u stringu tražilice odvađa riječ od
    riječi
    $delimiters = ",. ; ";
    /* na prvi poziv funkcije `strtok` isporučujete cijeli
    string tražilice
    i moguće znakove za odvajanje. Dobijete nazad prvu riječ
    stringa */
    $sword = strtok($searchString, $delimiters);
```

Među konstantama je i konstanta define("FT_MIN_WORD_LEN",4); To znači da riječi s 3 i manje znakova, neće biti tražene. Bit će ignorirane. Formiraju se arrayi prihvaćenih i ignoriranih riječi:

```
$accepted_words = array();
$ignored_words = array();
// petlja za učitavanje svih riječi, riječ po riječ
while ($sword)
{
    // kratke riječi se dodjeljuju jednom arrayu, a
    ostale riječi drugom arrayu
    if (strlen($sword) < FT_MIN_WORD_LEN)
        $ignored_words[] = $sword;
```

```

        else
            $accepted_words[] = $word;
            // uzimanje sljedeće riječi u stringu tražilice
            $word = strtok($delimiters);
        }
        // ako postoji ijedna riječ...
        if (count($accepted_words))
        {
            // zovemo metodu u podatkovnom sloju, koja će nam
            vratiti rezultat
            $search_results->mJedinica = $this->mPodatkovniObjekt-
>Search
            ($accepted_words, $allWords, $pageNo, $rTotalPages);
        }
        // pohrani liste prihvaćenih i zanemarenih riječi, ako
        postoje
        if ($accepted_words != null)
            $search_results->mSearchedWords = implode(", ",
            $accepted_words);
        if ($ignored_words != null)
            $search_results->mIgnoredWords = implode(", ",
            $ignored_words);
        // izbaci rezultat
        return $search_results;
    }

```

podatkovni_objekt.php

Dio kôda koji se odnosi na tražilicu:

```

    public function Search($words, $allWords, $pageNo,
    &$rTotalPages)
    {
        // ako je uključena opcija $allWords (`Sve riječi
        zastupljene`)
        // tad dodajemo "+" ispred svake riječi u arrayu $words
        if (strcmp($allWords, "on") == 0)
            for ($i = 0; $i < count($words); $i++)
                $words[$i] = "+" . $words[$i];
        // od arraya $words kreiraj jedan `search string`
        $temp_string = $words[0];
        for ($i = 1; $i < count($words); $i++)
            $temp_string = $temp_string . " $words[$i]";
        // izgradi upit za `search`
        if (strcmp($allWords, "on") == 0)
            // izgradi upit za traženje `sve riječi zastupljene`
            (all-words)
        $query_string = "SELECT jedinica_id, ime, CONCAT(LEFT(opis, ".
        BROJ_SLOVA_KRATKOG_OPISA . "),'...') AS opis, slika_1 FROM
        jedinica
        WHERE MATCH (ime,opis) AGAINST (\"$temp_string\" IN BOOLEAN
        MODE) ORDER BY MATCH (ime,opis)
        AGAINST (\"$temp_string\" IN BOOLEAN MODE)";
        else
            // izgradi upit za traženje `bilo koja riječ zastupljena`
            (any-words)

```

```

$query_string = "SELECT jedinica_id, ime,
CONCAT(LEFT(opis, ". BROJ_SLOVA_KRATKOG_OPISA .
"), '...') AS opis, slika_1 FROM jedinica WHERE MATCH
(ime, opis) AGAINST (\"$temp_string\");
// pozovi CreateSubpageQuery za aktiviranje prijeloma teksta
$page_query = $this->CreateSubpageQuery($query_string,
$pageNo, $rTotalPages);
// izvrši upit i izbacij rezultat
return $this->bPodataka->BpGetAll($page_query);
}

```

Vježba 9

SQL upiti u tražilici

U funkciji search u podatkovnom objektu imate ovaj SQL upit:

```

SELECT jedinica_id, ime, CONCAT(LEFT(opis, ".
BROJ_SLOVA_KRATKOG_OPISA . "), '...') AS opis, slika_1 FROM
jedinica
WHERE MATCH (ime, opis) AGAINST (\"$temp_string\" IN BOOLEAN
MODE) ORDER BY MATCH (ime, opis)
AGAINST (\"$temp_string\" IN BOOLEAN MODE);

```

Napišite isti upit u MySQL naredbodavnom retku, s konkretnim vrijednostima umjesto varijabli i konstante:

```

SELECT jedinica_id, ime, CONCAT(LEFT(opis, "4"), '...') AS
opis, slika_1 FROM jedinica
WHERE MATCH (ime, opis) AGAINST ("književnost" IN BOOLEAN
MODE) ORDER BY MATCH (ime, opis)
AGAINST ("književnost" IN BOOLEAN MODE);

```

```

C:\Program Files\MySQL\MySQL Server 4.1\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.1.14-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use prazna
Database changed
mysql> select jedinica_id, ime, CONCAT(left(opis, '4'), '...') as opis, slika_1 f
rom jedinica where match (ime, opis) against ('književnost' in boolean mode) orde
r by match (ime, opis) against ('književnost' in boolean mode);
+-----+-----+-----+-----+
| jedinica_id | ime           | opis      | slika_1 |
+-----+-----+-----+-----+
| 1           | Književnost 16. stolješa | Ok...    | krka1.jpg |
| 2           | Književnost 17. stolješa | Okl...    | roska1.jpg |
| 3           | Književnost 18. stolješa | Okls...   | visovac1.jpg |
| 4           | Književnost 19. stolješa | Okls...   | skradin1.jpg |
| 5           | Književnost 20. stolješa | Okls...   | st1.jpg |
| 6           | Književnost 21. stolješa | Okls...   | skradin2.jpg |
+-----+-----+-----+-----+
6 rows in set (0.03 sec)

mysql>

```

Objašnjenje tražilice

SQL upit za FULL-TEXT ALL-WORDS tražilicu glasi:

```

SELECT jedinica_id, ime, CONCAT(LEFT(opis, "4"), '...') AS
opis, slika_1 FROM jedinica

```

```
WHERE MATCH (ime, opis) AGAINST ("književnost" IN BOOLEAN
MODE) ORDER BY MATCH (ime, opis)
AGAINST ("književnost" IN BOOLEAN MODE);
```

Stupci ime i opis moraju biti indeksirani za FULL-TEXT tražilicu. To smo već ranije učinili pomoću ovog SQL-a:

```
ALTER TABLE jedinica ADD FULLTEXT ImeOpisFTIndex(ime, opis);
```

Relevantna lista koju je izbacila FULL-TEXT tražilica je složena po relevantnosti. Bolje su pozicionirani oni rezultati u kojima su pronađeni svi traženi izrazi, ili se pojavljuju više puta.

CONCAT(LEFT(opis, "4"

Pomoću SQL funkcija CONCAT i LEFT smo uzeli prva 4 slova mjesta iz stupca OPIS.

Zadatak

1. Unesite u aplikaciju nove jedinice koje u stupcu IME i OPIS sadrže iste riječi ali ne i u istom broju.
2. Ponovite SQL upit iz vježbe, ali sad za te nove riječi i provjerite da li je na vrhu liste jedinica u kojoj su sve riječi zastupljene i najviše puta.

SQL upit za FULL-TEXT ALL-WORDS tražilicu glasi:

```
SELECT jedinica_id, ime, CONCAT(LEFT(opis, "4"), '...') AS
opis, slika_1
FROM jedinica
WHERE MATCH (ime, opis)
AGAINST ("književnost 16. stoljeća";
```



```
mysql> select jedinica_id, ime, concat(left(opis, "4"), '...') as opis, slika_1
from jedinica where match (ime, opis) against ("marko marulić");
```

jedinica_id	ime	opis	slika_1
10	Marko	Mark...	generic_image_1.jpg
11	Marko	[Tip...	generic_image_1.jpg

```
2 rows in set (0.00 sec)

mysql> _
```

Zadatak

1. Unesite u aplikaciju nove jedinice koje u stupcu IME i OPIS sadrže iste riječi ali ne i u istom broju.
2. Ponovite SQL upit FULL-TEXT ALL-WORDS, ali sad za te nove riječi i provjerite da li je na vrhu liste jedinica u kojoj su sve riječi zastupljene i najviše puta.

Vježba 10

SQL Join

U podatkovnom objektu imamo i SQL s ključnom riječi JOIN:

```

    public function GetJedinicaUCjelini($cjelinaId, $pageNo,
    &$rTotalPages)
    {
        $query_string =
        "SELECT jedinica.jedinica_id, jedinica.ime,
        CONCAT(LEFT(opis," . BROJ_SLOVA_KRATKOG_OPISA . "), '...')
        AS opis,
        jedinica.slika_1,
        jedinica.promidzba_na_razini_podrucja,
        jedinica.promidzba_na_razini_homepage
        FROM jedinica INNER JOIN cjelina_jedinica
        ON jedinica.jedinica_id = cjelina_jedinica.jedinica_id
        WHERE cjelina_jedinica.cjelina_id = $cjelinaId";
        $page_query = $this->CreateSubpageQuery($query_string,
        $pageNo, $rTotalPages);
        return $this->bPodataka->BpGetAll($page_query);
    }

```

U MySQL naredbodavnom retku napišimo ovaj SQL:

```

SELECT jedinica.jedinica_id, jedinica.ime,
CONCAT(LEFT(opis,"4"), '...')
AS opis,
jedinica.slika_1,
jedinica.promidzba_na_razini_podrucja,
jedinica.promidzba_na_razini_homepage
FROM jedinica INNER JOIN cjelina_jedinica
ON jedinica.jedinica_id = cjelina_jedinica.jedinica_id
WHERE cjelina_jedinica.cjelina_id = 1;

```

Rezultat su SVE jedinice koje pripadaju CJELINI koja ima podatak cjelina_id = 1.

JOIN koristimo za SQL UPITE nad više tablica. U ovom UPITU su sudjelovale tri tablice: jedinica, cjelina, cjelina_jedinica.

Vježba 11

Administracija CMS

Administrator se logira preko forme u kojoj treba upisati korisničko ime i zaporku. U našem slučaju: admin i admin.

konfiguracija.inc.php

```

define("ADMIN_KORISNICKOIME", "admin");
define("ADMIN_ZAPORKA", "admin");

```

admin.php

```

<?php
require_once 'include/vrh_aplikacije.php';
require_once SITE_ROOT .
'/poslovni_objekti/poslovni_objekt.php';
// Ako administrator nije logiran, preusmjeri ga na stranicu
za logiranje login.php
// s dodanom varijablom ZelimOvuStranicu u GET stringu
// da se zna prema kud će se admin usmjeriti kad se logira

```

```

if (!(isset($_SESSION['AdministratorJeLogiran'])) ||
$_SESSION['AdministratorJeLogiran'] !=true)
{
    header('Location: login.php?ZelimOvuStranicu=admin.php');
    exit;
}
// Ako se administrator odjavi ...
if (isset($_GET['Stranica']) && $_GET['Stranica'] ==
"Odjava")
{
    unset($_SESSION['AdministratorJeLogiran']);
    header("Location: index.php");
    exit;
}
// Učitaj web stranicu instanciranjem objekta home
$home = new HomePage();
$sadrzajStranice = "blank.tpl";
// Ako u upitu nema varijable `Stranica`
// (drugim rijecima, ako Stranica nije eksplicitno zadana),
// podrazumjeva se stranica Podrucja
if(isset($_GET['Stranica']))
    $admin_stranica = $_GET['Stranica'];
else
    $admin_stranica = "Podrucja";
// Odabir stranice koja ce se ucitati ...
if ($admin_stranica == "Podrucja")
    $sadrzajStranice = "admin_podrucja.tpl";
if ($admin_stranica == "Cjeline")
    $sadrzajStranice = "admin_cjeline.tpl";
if ($admin_stranica == "Jedinice")
    $sadrzajStranice = "admin_jedinice.tpl";
if ($admin_stranica == "JedinicaDetalji")
    $sadrzajStranice = "admin_jedinica_detalji.tpl";

$home->assign("sadrzajStranice", $sadrzajStranice);
$home->display('admin.tpl');
require_once 'include/dno_aplikacije.php';
?>

```

login.php

```

<?php
require_once 'include/vrh_aplikacije.php';
$home = new HomePage();
$admin_login = new Login();
$home->assign_by_ref("admin_login", $admin_login);
$home->display('login.tpl');
require_once 'include/dno_aplikacije.php';
class Login
{
    public $mKorisnickoIme;
    public $mLoginPoruka = "";
    public $mZelimOvuStranicu;

    function __construct()
    {

```



```

        // Ako je zaporka ispravna, admin se preusmjerava na ovu
stranicu
        if (isset($_GET['ZelimOvuStranicu']))
            $this->mZelimOvuStranicu = $_GET['ZelimOvuStranicu'];
        if (isset($_POST['korisnickoime']))
            $this->mKorisnickoIme = $_POST['korisnickoime'];
        // Ako je admin vec logiran, preusmjerava se na trazenu
stranicu
        if (isset($_SESSION['AdministratorJeLogiran'])
            && $_SESSION['AdministratorJeLogiran'] == true)
        {
            header('Location: ' . $this->mZelimOvuStranicu);
            exit;
        }
        // Provjera korisnickog imena i zaporka
        // Kreiranje sesije i davanje vrijednosti true sesiji
        if(isset($_POST['Submit']))
        {
            if($_POST['korisnickoime'] == ADMIN_KORISNICKOIME
                && $_POST['password'] == ADMIN_ZAPORKA)
            {
                $_SESSION['AdministratorJeLogiran'] = true;
                header("Location: " . $this->mZelimOvuStranicu);
                exit;
            }
            else
                $this->mLoginPoruka = "<br />Login nije uspio. Molim
pokušajte ponovo:";
        }
    }
}
?>

```

templates

U mapi cms_komplet se nalazi kompletan kôd cms aplikacije i administracije.

U mapi tempaltes potrebno je kreirati ove smarty predloške:

admin.tpl

admin_cjeline.tpl

admin_jedinica_detalji.tpl

admin_jedinice.tpl

admin_podrucja.tpl

Kôd je objašnjen u komentarima.

smarty_plugins

Svaki od smarty predložaka ima i svoj drugi dio:

function.load_admin_cjeline.php

function.load_admin_jedinica_detalji.php

function.load_admin_jedinice.php

function.load_admin_podrucja.php

Kôd je objašnjen u komentarima.

Sažetak

CMS je profesionalna web aplikacija na kojoj smo naučili:

1. Objektno orijentirano programiranje.
2. Aplikacija u tri sloja: prezentacijski, poslovni i podatkovni sloj.
3. Korištenje Smarty paketa za razdvajanje XHTML-a od PHP.
4. Korištenje PEAR paketa za spajanje na bazu podataka.



E-trgovina

Metodika-----

Cilj ove nastavne jedinice je prikazati koliko su web aplikacije bliske jedna drugoj. CMS Ivamar aplikacija je ovdje transformirana u E-trgovinu Ivamar.

Cijela promjena se svodi na ugradnju stupca 'cijena' u tablicu i ugradnju sustava autorizacije i naplate.

Sve rečeno za CMS Ivamar vrijedi i za ovu aplikaciju.

Udžbenik-----

Što ćemo naučiti?

1. Baza podataka za e-trgovinu.
2. PHP Smarty PEAR kôd.
3. Cijena.
4. Dodaj u košaricu.
5. Pogled u košaricu.
6. Autorizacija prije naplate.

Baza podataka

Kompletna e-trgovina, PHP kôd i MySQL baza podataka se nalaze u mapi 'Poglavlje_e_trgovina'.

Baza podatka 'trgovina' se razlikuje od baze za cms samo u tome što sam tablici 'jedinica' dodao polje 'cijena':

```
`cijena` decimal(10,2) NOT NULL default '0.00'
```

trgovina.sql

```
DROP DATABASE IF EXISTS `trgovina`;  
  
CREATE DATABASE trgovina;  
  
GRANT ALL PRIVILEGES ON trgovina.* TO admin@localhost  
IDENTIFIED BY 'admin' WITH GRANT OPTION;  
  
USE trgovina;  
  
DROP TABLE IF EXISTS `podrucje`;
```

```

CREATE TABLE podrucje (
    podrucje_id int(11) NOT NULL auto_increment,
    ime varchar(50) NOT NULL default '',
    opis TEXT default NULL,
    PRIMARY KEY (podrucje_id)
) TYPE=MyISAM AUTO_INCREMENT=5 ;

DROP TABLE IF EXISTS `cjelina`;

CREATE TABLE `cjelina` (
    `cjelina_id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
    `podrucje_id` INT UNSIGNED NOT NULL ,
    `ime` VARCHAR( 50 ) NOT NULL ,
    `opis` TEXT ,
    PRIMARY KEY ( `cjelina_id` )
) TYPE=MyISAM;

DROP TABLE IF EXISTS `jedinica`;

CREATE TABLE `jedinica` (
    `jedinica_id` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
    `ime` VARCHAR( 50 ) NOT NULL ,
    `opis` TEXT NOT NULL ,
    `cijena` decimal(10,2) NOT NULL default '0.00',
    `slika_1` VARCHAR( 50 ) ,
    `slika_2` VARCHAR( 50 ) ,
    `promidzba_na_razini_homepage` VARCHAR( 1 ) NOT NULL ,
    `promidzba_na_razini_podrucja` VARCHAR( 1 ) NOT NULL ,
    PRIMARY KEY ( `jedinica_id` )
) TYPE=MyISAM;

ALTER TABLE jedinica ADD FULLTEXT ImeOpisFTIndex(ime,opis);

DROP TABLE IF EXISTS `cjelina_jedinica`;

CREATE TABLE `cjelina_jedinica` (
    `cjelina_id` INT UNSIGNED NOT NULL ,
    `jedinica_id` INT UNSIGNED NOT NULL ,
    PRIMARY KEY ( `cjelina_id` , `jedinica_id` )
) TYPE=MyISAM;

```

PHP Smarty PEAR kôd

Programski kôd je isti kao i kôd za CMS. Polje 'cijena' je dodano u sve SQL upite u podatkovnom objektu, vezane za tablicu 'jedinica'. Cijena je dodana i u parametre funkcija gdje je to bilo potrebno.

Plaćanje

Koristio sam PayPal firmu kao primjer vanjske usluge košarice, autorizacije kupca i naplate. To je samo primjer programskog kôda i gdje ga napisati, jer u trenutku

pisanja ovog teksta, firma PayPal još nije pružala usluge vlasnicima e-trgovina u Hrvatskoj. Postoje mnoge firme koje tu uslugu pružaju.

Promjene koje je potrebno učiniti u postojećem kôdu cms-a

index.tpl

U **sekciji HEAD** ćete dodati JavaScript kôd za otvaranje posebnog prozora za uslugu košarice i autorizacije.

Promjena je upisana između Smarty tagova `{literal}` i `{/literal}`. To je potrebno zato što Smarty koristi vitičaste zagrade za svoj kôd, a i JavaScript koristi vitičaste zagrade. Unutar tagova `{literal}` i `{/literal}`, Smarty neće interpretirati vitičaste zagrade kao svoje.

```
<head>
<title>{#naslov_title#}</title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250" />
<link rel="stylesheet" href="stil.css" type="text/css" />

{literal}
<script language="JavaScript" type="text/javascript">
<!--
var PayPalWindow = null;
function OpenPayPalWindow(url)
{
    if ((!PayPalWindow) || PayPalWindow.closed)
        // Ako PayPal window nije otvoren, otvorit ćemo ga
        PayPalWindow = window.open(url,"cart","height=300,
width=500");
    else
    {
        // <ako je PayPal window već otvoren, stavit ćemo ga
u fokus
        PayPalWindow.location.href = url;
        PayPalWindow.focus();
    }
}
// -->
</script>
{/literal}
</head>
```

U **sekciji BODY** ćete dodati kôd "Sadržaj košarice".

```
<a href=
"JavaScript:

OpenPayPalWindow('https://www.paypal.com/cgi-
bin/webscr?cmd=_cart&business=mojmail@mojmail.
com&display=1&return=www.mojbsite.com&cancel_retur
n=www.yourwebsite.com') ">
```

```

                
            </a>

```

Dodaj u košaricu

jedinica.tpl

Dodajte link "Dodaj u košaricu":

```

Cijena:
{$jedinica->mJedinica.cijena}
<br /><br />
<a href=
"JavaScript: OpenPayPalWindow('https://www.paypal.com/cgi-
bin/webscr?cmd=_cart&business=mojmail@mojmail.com&ite
m_name=' + escape('{$jedinica->mJedinica.ime}')
+'&amount={$jedinica-
>mJedinica.cijena}&add=1&return=www.mojsite.com&cancel_
return=www.mojsite.com')">


```

jedinica_kratko.tpl

Dodajte link "Dodaj u košaricu":

```

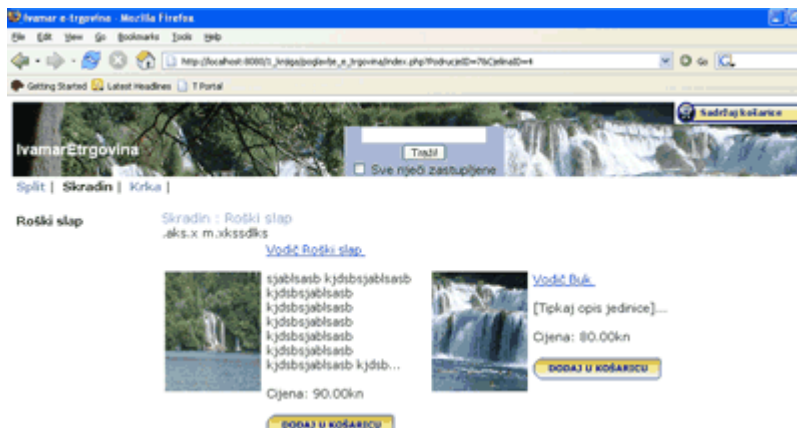
                <a href=
                    "JavaScript:
OpenPayPalWindow('https://www.paypal.com/cgi-
bin/webscr?cmd=_cart&business=mojmail@mojmail.com&ite
m_name=' + escape('{$lista_jedinica->mJedinica[k].ime}') +
'&amount={$lista_jedinica-
>mJedinica[k].cijena}&add=1&return=www.mojsite.com&am
p;cancel_return=www.mojsite.com')">
                    
                </a>

```

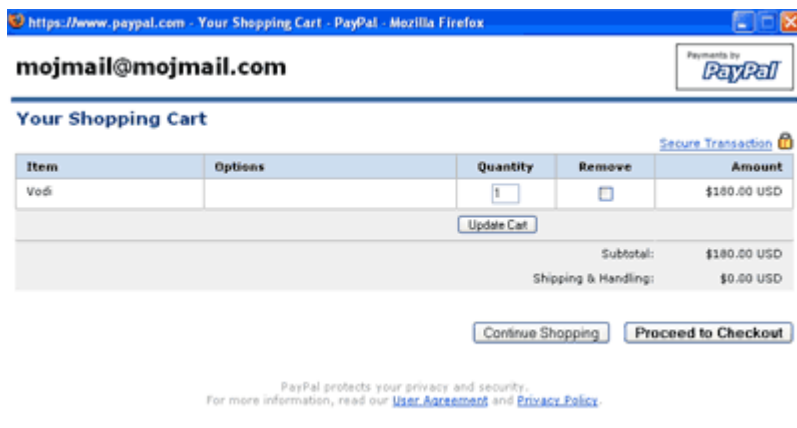
Administracija e-trgovine

Administracija je ista kao i za CMS. Na početnoj stranici je link "Administracija ovog site". Korisničko ime je 'admin' i zaporka je 'admin'.

E-trgovina u radu



Dodaj u košaricu.



ContinueShopping

Sadržaj košarice

Proceed to Checkout

https://www.paypal.com : Billing Information : PayPal : Mozilla Firefox

mojmail@mojmail.com

Payments by **PayPal**

Billing and Shipping Review Done

Billing Information

Signing up for a PayPal account will be required to complete this purchase.
* indicates required fields

Secure Transaction

Credit or Debit Card Information

*Country: United States
*First Name: (as it appears on card)
*Last Name: (as it appears on card)
*Payment Type:
*Credit Card Number:
*Expiration Date: 01 2006

Billing Address

*Address 1:
Address 2:
*City:
*State:
*ZIP Code: (5 or 9 digits)
*Is this your shipping address? ☒ Yes, it is the same as my shipping address ☐ No

Pay With PayPal

PayPal
Email Address
[Forgot your email address?](#)
Password
[Forgot your password?](#)

Order Summary

Subtotal:	\$270.00 USD
Shipping:	\$0.00 USD
Order Total:	\$270.00 USD

[View Details](#)

Contact Information and Security Check

*Email Address:

Sažetak

Cijeli kôd e-trgovine je objašnjen u prethodnoj nastavnoj cjelini. Isti je kao za CMS. Kôd koji trebate ugraditi u svoju e-trgovinu za davatelje usluga naplate preko Interneta je jednostavan. Vezan je za linkove Dodaj u košaricu i Sadržaj košarice. U primjeru smo koristili naš e-katalog s cijenama i navedenim linkovima, a košaricu, autorizaciju i naplatu izvana.

Literatura

1. Beginning PHP 5 and MySQL E-Commerce From Novice to Professional, Cristian Darie and Mihai Bucica, Apress 2005.
2. www.php.net
3. www.mysql.org
4. <http://smarty.php.net/>
5. <http://pear.php.net/>
6. <http://www.apachefriends.org/en/xampp.html>