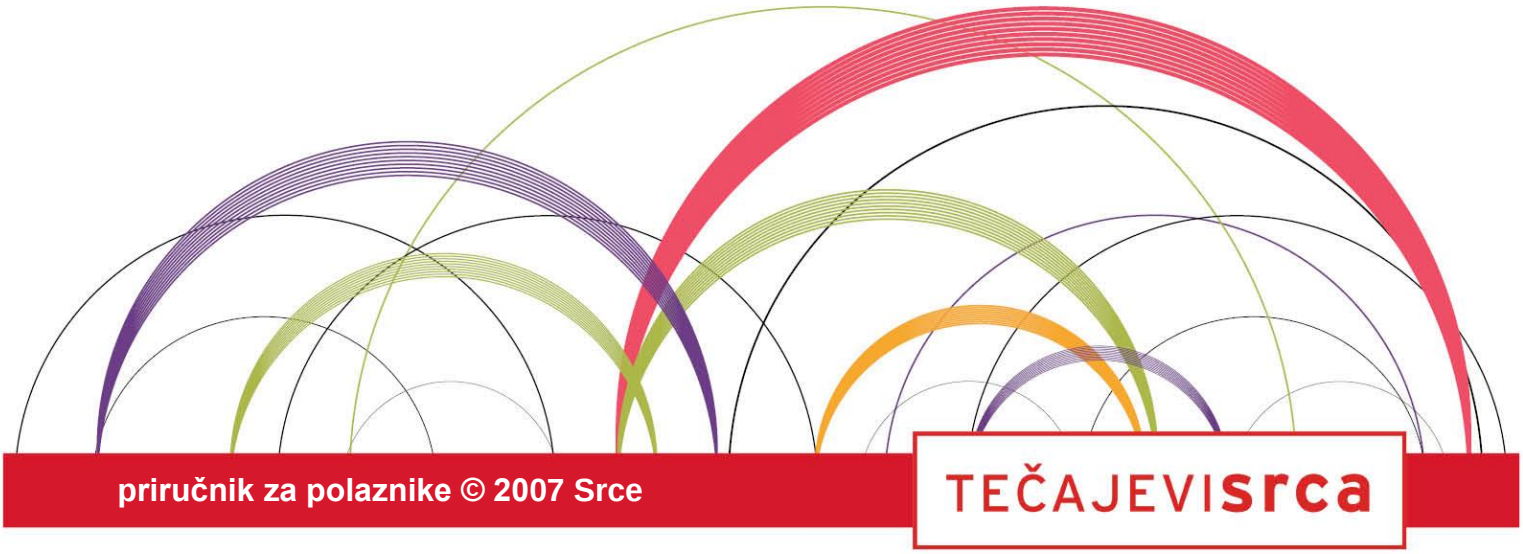
Uvod u PHP i MySQL

D350



priručnik za polaznike © 2007 Srce

TEČAJEVIsrca

srce
Sveučilište u Zagrebu
Sveučilišni računski centar

Ovu inačicu priručnika izradio je autorski tim Srca u sastavu:

Autor: Edin Mujadžević

Recenzent: Kruno Golubić

Urednica: Sabina Rako

Lektorica: Marijana Češi

Sveučilište u Zagrebu

Sveučilišni računski centar

Josipa Marohnića 5, 10000 Zagreb

tecajevi@srce.hr

ISBN: 978-953-7138-50-9 (meki uvez)
ISBN: 978-953-7138-51-6 (PDF)

Verzija priručnika D350-20141204

Sadržaj

Uvod

Svrha ovog tečaja je upoznavanje s programskim jezikom za dinamičko stvaranje web stranica PHP i sustavom za upravljanje relacijskim bazama podataka MySQL.

Za svladavanje tečaja potrebno je poznavanje osnova jezika za označavanje HTML, koji služi za izradu web stranica. Poželjno je i poznavanje osnova jezika za postavljanje upita bazama podataka SQL. Također je poželjno poznavanje osnova programiranja, ali nije nužno za uspješno svladavanje tečaja.

Priručnik se sastoji od 14 poglavlja koja se obrađuju pet dana, pet sati dnevno, koliko traje tečaj. Na kraju svakog poglavlja nalaze se vježbe koje će polaznici izrađivati zajedno s predavačem. Mogući savjeti i zanimljivosti istaknuti su u okvirima sa strane.

Polaznici će naučiti kako se programira u jeziku PHP, te kako se pomoću njega mogu generirati web stranice. Također će biti obrađeno dohvaćanje i spremanje podataka u bazu podataka, što je osnova na kojoj se temelje dinamičke i interaktivne web stranice i aplikacije.

Tijekom tečaja bit će, uz jednostavne vježbe, izrađen i praktični primjera koji će rabiti bazu podataka za spremanje podataka – jednostavni adresar s mogućnošću slanja poruka elektroničke pošte. U dodatku tečaja dan je i primjer jednostavnog sustava za upravljanje sadržajem web stranica (CMS), koji korisnici mogu iskoristiti za vježbu kod kuće.

Ovaj tečaj može poslužiti kao nastavak tečaja "Uvod u HTML", koji osigurava potrebno poznavanje osnova jezika HTML. Za daljnje upoznavanje s web tehnologijama nakon ovog tečaja polaznicima se može preporučiti tečaj "Osnove JavaScripta" i neki od tečajeva koji obrađuju objavu web stranica na internetskim poslužiteljima, no spomenuti tečajevi mogu se odslušati i prije slušanja ovog tečaja, budući da nisu izravno povezani s njim. Za nastavak proučavanja jezika SQL i relacijskih baza podataka preporuča se tečaj "Uvod u SQL".

Za izradu vježbi predviđena je mapa **"..\D350\vjezbe"**, u kojoj se nalaze sve potrebne datoteke. Gotova rješenja svih vježbi nalaze se u mapi **"..\D350\rjesenja"**. U mapi **"..\D350\primjeri"** nalaze se primjeri korišteni u poglavljima, pa ih je moguće i isprobati.

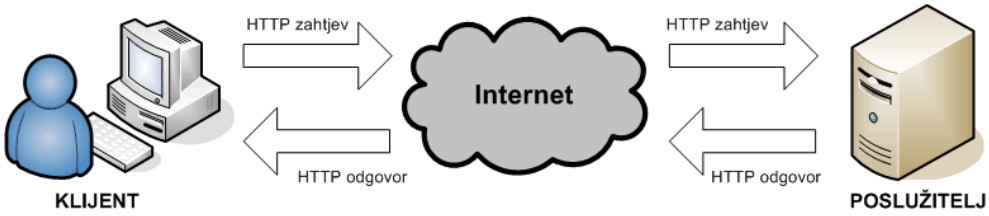
1. Uvod u PHP i MySQL

PHP je skriptni jezik koji se izvršava na poslužitelju, a glavna mu je namjena dinamičko stvaranje web stranica. MySQL je sustav za upravljanje relacijskim bazama podataka, i zajedno s programskim jezikom PHP predstavlja jedno od najpopularnijih rješenja za izradu dinamičkih web stranica i web aplikacija temeljenih na bazama podataka.

1.1. Klijentsko i poslužiteljsko skriptiranje

U komunikaciji putem Interneta sudjeluju dvije strane: klijent i poslužitelj (*server*). Klijent (korisnikov web preglednik) šalje zahtjev, npr. adresu neke stranice, a poslužitelj isporučuje odgovor, koji sadrži HTML kôd tražene stranice (i zajedno s njim binarne datoteke, npr. slike, video...), što će klijent interpretirati i prikazati korisniku kao web stranicu.

Komunikacija između poslužitelja i klijenta odvija se putem HTTP protokola, koji se sastoji od HTTP zahtjeva i odgovora. Komunikacija preko Interneta je dvosmjerna – i klijent može unutar HTTP zahtjeva poslati neke podatke poslužitelju.



Većina jezika koji se koriste u web programiranju (PHP, JavaScript, ASP i drugi) podskup su **skriptnih jezika** za koje je karakteristično da se program ne čuva u izvršnoj (*executable*) datoteci (koja je prevedena samo jednom), već se prevođenje naredbi obavlja pri svakom izvršavanju programa. Datoteke i dijelovi kôda napisani u takvim jezicima nazivaju se skriptama.

Statičke HTML stranice

Sa statičkim HTML stranicama, web stranica će uvijek biti ista, a mogućnosti joj se svode na prikazivanje teksta i slika, s mogućnošću povezivanja, preko linkova, s drugim stranicama.

HTML stranice s klijentskim skriptama

Klijentske skripte se zajedno s HTML kôdom (unutar HTML datoteka ili unutar zasebnih datoteka) nalaze na poslužitelju. Kad stigne zahtjev, zajedno s HTML kôdom se šalju klijentu, a na klijentu se izvršavaju kad budu pozvane (nakon što se stranica učita u preglednik, kad korisnik klikne na dugme i slično). Klijentske skripte nude veliko proširenje mogućnosti web stranica, od kojih je možda najkorisnije prikazivanje prozora s porukama, trenutno prikazivanje i skrivanje određenih dijelova na stranici i drugi dinamični vizualni efekti (promjena slike pri prelasku mišem i slično).

HTML stranice stvorene pomoću poslužiteljskih skripti

Poslužiteljske skripte, kao i klijentske skripte, nalaze se na poslužitelju i, također, mogu biti ubačene u HTML kôd, ili se mogu nalaziti u zasebnim datotekama. Kad stigne zahtjev od klijenta, skripta se izvršava na poslužitelju, a kao rezultat izvršavanje dobiva se HTML kôd koji se šalje klijentu. Najveća prednost poslužiteljskih skripti je njihova mogućnost

povezivanja s bazama podataka, što je omogućilo pojavu web stranica i web aplikacija s dinamičkim sadržajem koji se čita iz baze podataka i koji korisnici mogu mijenjati preko web aplikacije.

Usporedni pregled značajki i mogućnosti klijentskih i poslužiteljskih skripti dan je u ovoj tablici:

Klijentske skripte	Poslužiteljske skripte
izvršavaju se na klijentu	izvršavaju se na poslužitelju
nekompatibilnost između preglednika (neke stvari neće raditi u potpunosti u svim preglednicima)	aplikacija će uvijek raditi na isti način, neovisno o pregledniku
nemogućnost (izravnog) povezivanja s bazama podataka	povezivanje s bazama podataka
mogućnost reagiranja na velik broj događaja kao što je, npr., povećanje prozora, desni klik miša i sl.	aplikacija može reagirati samo na klik na link ili na dugme
aplikacija reagira trenutačno, stranica se ne osvježava	dulje vrijeme čekanja na odgovor, potrebno osvježavanje cijele stranice
poslužitelj se rasterećuje korištenjem resursa klijenta	uvijek se koriste resursi poslužitelja

Klijentsko i poslužiteljsko skriptiranje nisu alternativa jedno drugome, odnosno, većinu stvari koje se može postići jednim načinom programiranja ne može se postići drugim te ih je najbolje koristiti u kombinaciji.

Glavni klijentski skriptni jezik je JavaScript (čija je sintaksa slična programskim jezicima C i Java), koji je podržan u najvećem dijelu web preglednika. Uz njega postoje još JScript (Microsoftova verzija JavaScripta) i VBScript (skriptni jezik sa sintaksom sličnom programskom jeziku Visual Basic, također Microsoftov).

AJAX (Asynchronous JavaScript and XML) je tehnologija koja kombinira klijentsko i poslužiteljsko skriptiranje. Na određenu korisnikovu akciju klijentska skripta šalje pozadinski zahtjev skripti na poslužitelju koja obavlja dohvaćanje podataka iz baze podataka i šalje ih korisnikovom pregledniku, bez osvježavanja cijele stranice. Upotrebom tehnologije AJAX web aplikacije postaju brže za uporabu i imaju bolje korisničko sučelje od aplikacija koje se temelje samo na poslužiteljskim tehnologijama.

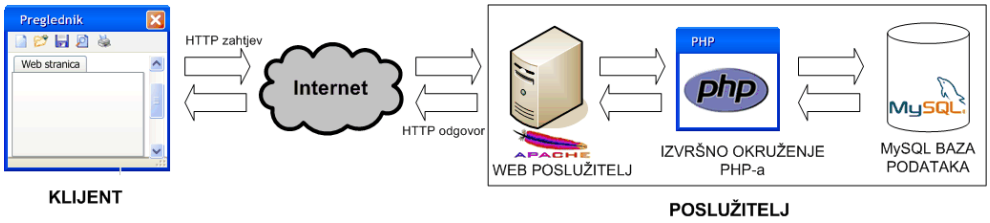
PHP: Hypertext Preprocessor je tzv. rekurzivna skraćenica, kod koje se sama skraćenica pojavljuje u značenju. U svijetu *open source* tehnologija su takve skraćenice česte, a druga poznata skraćenica je GNU (*GNU's Not Unix*).

1.2. PHP i ostale poslužiteljske tehnologije

Skriptni jezik PHP nastao je 1994. kao osobni projekt jednog entuzijasta, a kasnije se u njegov razvoj uključio veliki broj programera koji su svi doprinijeli razvoju jezika PHP. Originalno je skraćenica PHP značila *Personal Home Page Tools*, a kasnije je značenje promijenjeno i danas znači *PHP: Hypertext Preprocessor* (PHP: hipertekstualni pretprocesor, što opisuje glavnu funkciju jezika PHP - da na temelju PHP naredbi generira HTML, jezik kojim se opisuje hipertekst.)

Najčešći način instalacije PHP-a je u sklopu tzv. LAMP platforme (Linux, Apache, MySQL i PHP). Web poslužitelj Apache prima HTTP zahtjev, i ako tražena stranica ima nastavak **.php** (ili neki od drugih nastavaka koji su podešeni na poslužitelju), poziva izvršno okruženje PHP-a, koje prevodi i izvršava PHP naredbe. Ukoliko skripta treba dohvatiti neke podatke iz baze podataka, bazi se postavljaju upiti iz PHP-a. (Napomena: radi poboljšanja performansi, MySQL može biti instaliran na zasebnom poslužitelju.) Dobiveni podaci se prosljeđuju PHP-u, i PHP ih uklapa unutar HTML kôda. Taj HTML kôd se unutar HTTP odgovora šalje natrag klijentu i podaci se prikazuju u korisnikovom web pregledniku.

Posljednja verzija PHP-a je PHP 5.



Ostale poslužiteljske tehnologije

Dugo godina veliki konkurent skriptnom jeziku PHP bio je ASP (Active Server Pages), Microsoftova poslužiteljska tehnologija čije su mogućnosti bile podjednake. No, ASP je zastario, a PHP se nastavio razvijati i dobivati bolju podršku za objektno-orijentirano programiranje.

Postoji i nekoliko *frameworka* (razvojnih platformi) za PHP (Zend Framework, Symfony) čiji je cilj olakšati programiranje u njemu odvajanjem prezentacije (HTML) od logike (PHP kôd), te uvođenjem objekata koji predstavljaju HTML elemente i proširuju funkcionalnost web stranica, zatim objekata za rad s bazama podataka i drugih.

Veliki konkurent je Microsoftov ASP.NET, donekle nasljednik tehnologije ASP, koji koristi .NET Framework za koji je moguće razvijati u više .NET jezika, od kojih su najrašireniji C# i Visual Basic.NET. Njegova prednost je veliki broj gotovih kontrola i objekata, i to što se njegove datoteke ne moraju prevoditi svaki put kod izvršavanja, već samo prvi put.

Jaka konkurencija je i JSP (JavaServer Pages). Za JSP danas postoji velik broj *frameworka* (Spring, Struts, JavaServer Faces...). Kao i ASP.NET, ne mora se prevoditi svaki put kod izvršavanja, i vrlo je raširen u svijetu poslovnih aplikacija.

Perl je skriptni jezik UNIX sustava, i to jedan od prvih s kojima je započeo razvoj web aplikacija. Ima velike mogućnosti, a njegov razvoj i dalje traje pa se i danas koristi za razvoj web aplikacija.

Python je skriptni jezik koji se može upotrebljavati i za web. Za njega postoji nekoliko *frameworka* i zajednica entuzijasta koji ga koriste.

Ruby je jezik sličan jeziku Python, a u posljednje vrijeme je vrlo popularan u kombinaciji s Rails *frameworkom* koji omogućava brz i jednostavan razvoj aplikacija.

Jedan od konkurenata skriptnom jeziku PHP je i ColdFusion, poslužiteljski skriptni jezik i tehnologija u vlasništvu Macromedie.

1.3. MySQL i druge baze podataka

MySQL je najrasprostranjeniji *open source* sustav za upravljanje bazama podataka na webu. Njegov razvoj započeo je 1995. godine. Neki od korisnika sustava MySQL su YouTube, Filckr, Wikipedia i Wordpress.

Među ostalim *open source* sustavima za upravljanje bazama podataka treba spomenuti PostgreSQL,sustav koji se više koristi u poslovnim sustavima, a manje na webu.

Među komercijalnim sustavima za upravljanje bazama podataka najznačajniji su Microsoft SQL Server i Oracle Database. I jedan i drugi upotrebljavaju se u zahtjevnim poslovnim okruženjima, pri čemu se prvi značajno koristi i na webu.

Baze podataka napravljene u programu Microsoft Access se također često upotrebljavaju na webu, pogotovo u kombinaciji s ASP-om. No, treba napomenuti da su performanse takve baze podataka slabe u usporedbi s pravim sustavima za upravljanje bazama podataka.

1.4. Osnove PHP sintakse

Najbolje je upotrebljavati prvi način pisanja PHP oznaka jer skraćeni način pisanja ne mora biti podržan na svim poslužiteljima.

PHP naredbe najčešće se nalaze umetnute unutar HTML kôda zajedno s HTML oznakama. Da bi ih PHP-ov prevoditelj mogao prepoznati, moraju se nalaziti unutar posebnih oznaka. Najčešći i preporučeni način pisanja PHP oznaka je ovaj:

```
<?php

?>
```

Skraćeni način pisanja PHP oznaka izgleda ovako:

```
<?

?>
```

Svaka PHP naredba mora završiti sa znakom točka-zarez, inače će se prilikom izvršavanja javiti sintaksna greška. Primjer naredbe u jeziku PHP:

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

```
Dobar dan
```

Naredba ***echo*** ispisuje zadani niz znakova, pa će gornji primjer ispisati tekst "Dobar dan".

Moguće je uz naredbe pisati i pomoćni tekst (komentare) koji ne utječe na izvođenje programa, a služi u svrhu opisa napisanog kôda. Komentari koji se nalaze u jednoj liniji započinju znakovima ***//*** ili znakom #, a komentari koji se protežu kroz jednu ili više linija započinju znakovima ***/****, a završavaju znakovima ****/***.

```
<?php
    // naredba koja ispisuje tekst
    echo "Dobar dan";
?>
```

Drugi primjer je:

```
<?php
    /* naredba koja
       ispisuje tekst */

    echo "Dobar dan";
?>
```

Vježba 1.1 – Prva PHP skripta

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *index.php*.

2. Unutar stvorene datoteke stvorite kostur HTML dokumenta - dodajte oznake *head* i oznake *title* i *meta* unutar njega, a nakon njih oznake *body* i *h2*:

```
<html>
<head>
  <title>Prva PHP skripta</title>
</head>
<body>
  <h2>Prva PHP skripta</h2>

</body>
</html>
```

Na web poslužitelju s instaliranim prevoditeljem za PHP, početna stranica za svaku mapu je ona koja se zove *index.php*. U tom slučaju za pokretanje datoteke u web pregledniku nije potrebno navoditi punu putanju do datoteke, već je dovoljna putanja do mape.

3. Između oznake h2 i završne oznake body dodajete početnu i završnu oznaku za PHP kôd, a unutar njih sljedeću naredbu (masno otisnuti kôd):

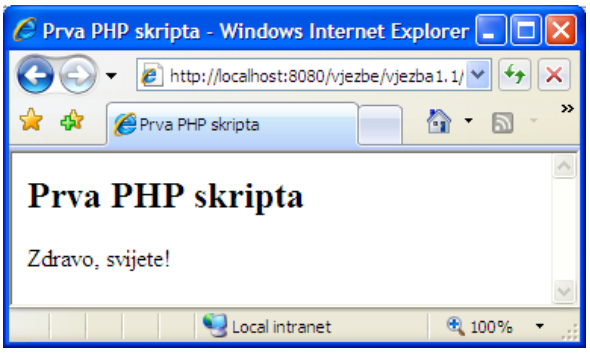
```
  <h2>Prva PHP skripta</h2>
  <?php
      echo "Pozdrav svijete!";
  ?>
</body>
```

Datoteku spremite u mapu "…\D350\vjezbe\vjezba1.1".

4. Pokrenite servis *EasyPHP*, ako nije već pokrenut.

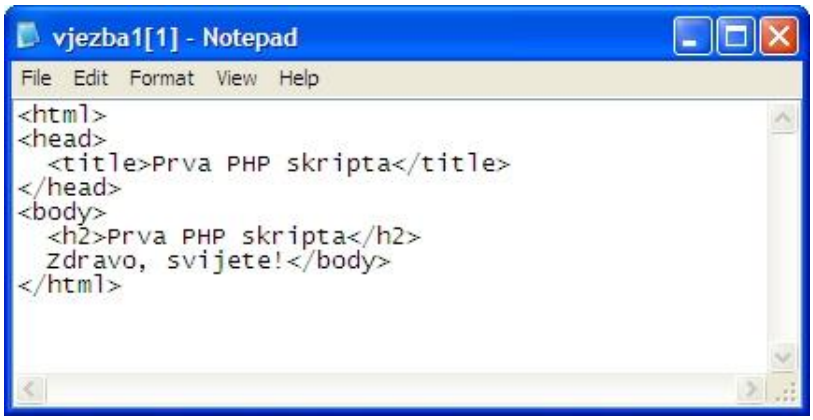
5. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba1.1/*

Ako ste nakon instalacije *EasyPHP*-a promijenili port iz 80 u neku drugu vrijednost, npr. 8080, upisat ćete adresu "http://localhost:8080/Adresar.htm"



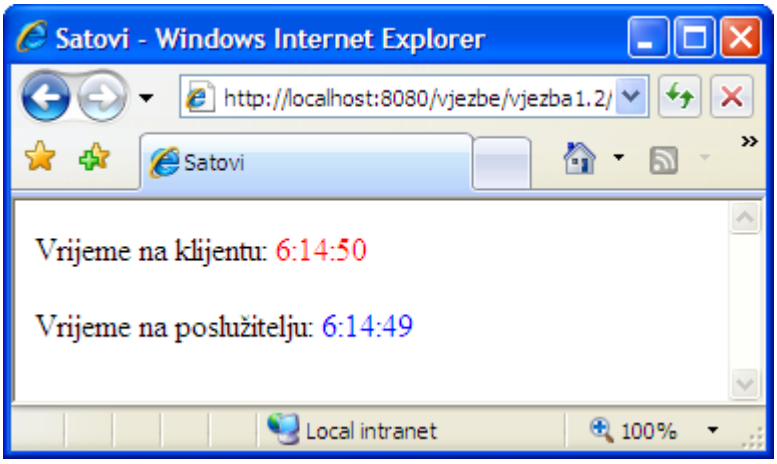
6. Uz naslov "Prva PHP skripta", u pregledniku je ispisan i tekst "Pozdrav svijete!".

7. Odaberite iz izbornika View→ Source. Pogledajte kôd stvorene HTML stranice. Primjećujete da u njemu nema PHP oznaka niti naredbe *echo*, već je tu samo tekst "Pozdrav svijete!". To znači da web preglednik dobiva već gotovi HTML kôd koji je generiran na poslužitelju kad je pozvana stranica *index.php*.



Vježba 1.2 – Razlika između klijentskog i poslužiteljskog skriptiranja

1. Pokrenite servis *EasyPHP*, ako nije već pokrenut.
2. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba1.2/*



3. Osvježite stranicu (klikom na dugme ili kombinacijom tipki Ctrl + F5).

   Prvi sat ispisuje vrijeme pomoću JavaScripta, i zato se vrijeme stalno pomiče, jer se skripta izvršava na strani klijenta i stalno ažurira vrijeme (u intervalima od 1 sekunde).

   Drugi sat je napravljen pomoću PHP-a, i stoga se ažurira samo kad se stranica osvježi (osvježavanje stranice je ponovno slanje zahtjeva poslužitelju i primanje njegovog odgovora).

   Svaki put kad se zahtjev ponovno pošalje poslužitelju, pomoću PHP-a se izračunava trenutno vrijeme, ispisuje unutar HTML-a, i šalje natrag klijentu.

U ovom je poglavlju obrađeno:

- PHP i MySQL
- klijentsko i poslužiteljsko skriptiranje
- kratak pregled poslužiteljskih tehnologija i baza podataka
- osnovna sintaksa PHP-a

2. Varijable i operacije nad njima

Svaki funkcionalni program mora čitati određene podatke i omogućavati obradu i pohranu podataka. Za privremeno spremanje podataka tijekom izvođenja programa koriste se **varijable.**

Za obradu i izračune nad podacima koriste se **operatori** koji su zapravo simboli za različite operacije (pridruživanje, aritmetičke operacije, operacije usporedbe i drugo).

2.1. Varijable

Varijable se u programskim jezicima koriste kao privremeni spremnici za pohranjivanje vrijednosti. Vrijednosti koje se spremaju u varijable mogu biti različitog tipa podataka, što znači da mogu biti tekstualne, brojčane, logičke ili drugog tipa.

PHP podržava sljedeće osnovne **tipove podataka**:

- logički (Booleov) tip podatka – *true*, *false*
- cjelobrojni – 1, 2, 123, -46…
- decimalni - 3.1459, -234.2045
- tekstualni (niz znakova) – "a", "Danas je lijep dan"

Složeni tipovi podataka u jeziku PHP su ovi:

- polje – podatak koji sadrži više povezanih vrijednosti
- objekt – složeni tip podatka koji predstavlja objekt iz stvarnog svijeta (može sadržavati podatke jednostavnog tipa, ali i funkcije)

Postoje i dva specijalna tipa:

- resource – vrijednost koju vraća neka funkcija (npr. memorijska adresa, datoteka, skup zapisa iz baze podataka)
- NULL – tip koji poprima varijabla kojoj nije pridružena vrijednost

PHP pripada skupini slabo tipiziranih jezika. Za razliku od većine drugih programskih jezika, varijablu u PHP-u prije uporabe nije potrebno deklarirati, odnosno nije potrebno navesti njen tip. Za uporabu varijable dovoljno je navesti njezino ime i pridružiti joj neku vrijednost. Varijabla će sama poprimiti odgovarajući tip podatka ovisno o pridruženoj vrijednosti.

Štoviše, moguće je tijekom uporabe varijable promijeniti njen tip, tako da joj pridružimo vrijednost koja je drugog podatkovnog tipa. Preporuča se ipak ne mijenjati tip varijable jer to često dovodi do pogrešaka u pisanju programa.

Ispred svake varijable u jeziku PHP navodi se znak ***$.*** Imena varijabli (identifikatori) mogu sadržavati slova, brojke ili podvlaku (_), a prvi znak (poslije ***$***, naravno) može biti samo slovo ili podvlaka. Primjeri ispravnih i neispravnih imena varijabli dani su u tablici:

ispravno	neispravno
$mojaVarijabla	*mojaVarijabla*
$varijabla1	*$1varijabla*
$moja_Varijabla	*$moja Varijabla*
$_varijabla	*$v@rijabla*

U imenima varijabli razlikuju se velika i mala slova, tako da *$mojaVarijabla* i *$mojavarijabla* neće biti prepoznate kao ista varijabla.

2.2. Konstante

Vrijednost spremljena u varijabli može se mijenjati tijekom izvođenja programa. **Konstante** također mogu sadržavati vrijednosti kao i varijable, ali se ta vrijednost ne može mijenjati jednom kada se deklarira.

Vrijednost se konstantama dodjeljuje preko funkcije *define* koja sadrži dva parametra: ime konstante i njezinu vrijednost.

Dodjela vrijednosti za tekstualnu vrijednost izgleda ovako:

```
<?php
  define('PI', 3.14);
?>
```

Ime konstante se obično piše velikim slovima, iako to nije obavezno, ali ovakav način označavanja omogućava lakše razlikovanje konstanti i varijabli. Također, prije imena konstante nije potrebno stavljati znak „$“. Time je olakšano i korištenje vrijednosti konstante koja se poziva samo imenom.

Jednom kada se konstanta definira, ona se u programu ne može promijeniti niti izbrisati.

2.3. Pridruživanje vrijednosti varijabli

Da bi varijabla *$varijabla* poprimila neku vrijednost potrebno je napisati sljedeće:

```
<?php
  $varijabla = vrijednost;
?>
```

Gornja naredba je naredba pridruživanja, a znak = je operator pridruživanja. S lijeve strane operatora pridruživanja uvijek se mora nalaziti varijabla, a s lijeve strane se može nalaziti konstantna vrijednost, rezultat nekog izraza (npr. aritmetičkog), ili pak poziv funkcije.

Dodjela vrijednosti za tekstualnu vrijednost izgleda ovako:

```
<?php
  $prvaVarijabla = "Ana";
  $drugaVarijabla = 'Ivo';
?>
```

Za zapisivanje znakovnih nizova, u jeziku PHP mogu se koristiti i jednostruki i dvostruki navodnici.

Dodjela vrijednosti za cjelobrojnu vrijednost izgleda ovako:

```
<?php
  $prvaVarijabla = 1;
  $drugaVarijabla = 2435;
?>
```

Dodjela vrijednosti decimalnog tipa izgleda ovako (koristi se decimalna točka, a ne zarez):

```
<?php
  $prvaVarijabla = 12.34;
  $drugaVarijabla = 0.05;
?>
```

Moguće je i korištenje eksponencijalnog zapisa:

```
<?php
  $prvaVarijabla = 1.2e23;
  $drugaVarijabla = 0.2E-10;
?>
```

Prva varijabla ima vrijednost $1.2 * 10^{23}$, dok druga ima vrijednost $0.2 * 10^{-10}$. Oznaka za eksponent (*e*) može se pisati i malim i velikim slovom.

Vrijednosti varijabli koje sadrže logički tip podataka također nisu osjetljiva na razliku između malih i velikih slova. Ovako izgleda dodjela vrijednosti varijabli logičkog tipa:

```
<?php
  $prvaVarijabla = True;
  $drugaVarijabla = FALSE;
?>
```

Dodjeljivanje vrijednosti varijabli može se izvršiti i na način da se varijabli pridruži vrijednost druge varijable:

```
<?php
  $prvaVarijabla = $drugaVarijabla;
?>
```

2.4. Aritmetički operatori

Osnovni aritmetički operatori

Nad varijablama brojčanog tipa podataka moguće je obavljati razne aritmetičke operacije. Pregled aritmetičkih operatora dostupnih u jeziku PHP dan je u tablici:

Operator	Značenje	Primjer
-	negativan predznak	`- $a`
+	zbrajanje	`$a + $b`
-	oduzimanje	`$a - $b`
*	množenje	`$a * $b`
/	dijeljenje	`$a / $b`
%	operacija modulo – (vraća ostatak pri dijeljenju)	`$a % $b`

Računanje zbroja vrijednosti dviju varijabli izgleda ovako:

```
<?php
  $zbroj = $a + $b;
?>
```

Na desnoj strani u naredbi pridruživanja može se naći i složeniji aritmetički izraz:

```
<?php
  $vrijednost = 2 + $a * $b;
?>
```

Prioritet aritmetičkih operacija

Prilikom izvršavanja, operacije se neće izvršavati redoslijedom kojim su napisane, već prema prioritetu aritmetičkih operacija. Prioritet osnovnih aritmetičkih operacija dan je u tablici:

Prioritet	Operatori	Značenje
1	-	negativan predznak
2	*, /, %	množenje, dijeljenje, modulo
3	+, -	zbrajanje, oduzimanje

Dakle, u gornjem primjeru najprije bi se izračunao umnožak vrijednosti varijabli *$a* i *$b*, a zatim bi se ta vrijednost zbrojila s 2.

Ukoliko želimo drugačiji redoslijed izvršavanja operacija, možemo koristiti zagrade, koje su korisne i radi povećavanja čitljivosti aritmetičkog izraza. Primjer dolje dao bi drugi rezultat od gornjeg primjera, jer bi se prvo izvršila operacija zbrajanja:

```
<?php
  $vrijednost = (2 + $a) * $b;
?>
```

Operatori uvećanja i umanjenja za 1

U jeziku PHP postoje i operatori uvećanja i umanjenja za 1. Pregled tih operatora dan je u tablici:

Operator	Značenje	Primjer
++	povećanje za 1	$a++
--	smanjenje za 1	$b--

Vrijednost varijable se može povećati za 1 na ovaj način:

```
<?php
  $broj = $broj + 1;
?>
```

Ovakva naredba može se zamijeniti s kraćim izrazom:

```
<?php
  $broj++;
?>
```

Složeni operatori pridruživanja

Uz jednostavan operator pridruživanja (=), postoje i složeni operatori pridruživanja. Varijabla se može povećati za neku vrijednost na ovaj način:

```
<?php
  $broj = $broj + 5;
?>
```

Ova se naredba može napisati jednostavnije pomoću operatora **+=** :

```
<?php
  $broj += 5;
?>
```

Potreban je velik oprez prilikom uporabe operatora ++ i -- u većim aritmetičkim izrazima.

Kod **prefiksnog oblika** pisanja operatora, ++$a, (u kojem operator dolazi prije varijable), povećavanje vrijednosti varijable za 1 izvršit će se prije ostatka izraza.

```
$a = 2;
$b = 1 + ++$a;
```

Gornje naredbe ekvivalentne su:

```
$a = 2;
$a = $a + 1;
$b = 1 + $a;
```

Kod **postfiksnog oblika**, $a++ (u kojem operator dolazi poslije varijable), povećanje vrijednosti varijable za 1 dogodit će se tek nakon što je izraz već izračunat, dakle neće imati utjecaja na izračunavanje izraza.

```
$a = 2;
$b = 1 + $a++;
```

Gornje naredbe ekvivalentne su:

```
$a = 2;
$b = 1 + $a;
$a = $a + 1;
```

Složeni operatori pridruživanja postoje za sve osnovne aritmetičke operacije:

Operator	Značenje	Primjer	Ekvivalentna naredba
+=	dodavanje vrijednosti s lijeve strane	`$a += vrijednost`	`$a = $a + vrijednost`
-=	oduzimanje vrijednosti s lijeve strane	`$a -= vrijednost`	`$a = $a - vrijednost`
*=	množenje s vrijednošću s lijeve strane	`$a *= vrijednost`	`$a = $a * vrijednost`
/=	dijeljenje s vrijednošću s lijeve strane	`$a /= vrijednost`	`$a = $a / vrijednost`
%=	operacija modulo s vrijednošću s lijeve strane	`$a %= vrijednost`	`$a = $a % vrijednost`

2.5. Spajanje znakovnih nizova

Više znakovnih nizova moguće je pretvoriti u jedan koristeći se operatorom spajanja. Kao operator spajanja koristi se znak **.** (točka).

```
<?php
  $niz1 = "Dobar";
  $niz2 = "dan";
  echo $niz1 . $niz2;
?>
```

```
Dobardan
```

Rezultat je ispisani tekst „Dobardan".

Ako se između riječi „dobar“ i riječi „dan“ želi umetnuti razmak, to se može učiniti ovako:

```
<?php
  echo $niz1 . " " . $niz2;
?>
```

```
Dobar dan
```

I za operator **.** može se koristiti skraćeni način pridruživanja:

```
<?php
  $niz = "Dobar";
  $niz .= "dan";
  echo $niz;
?>
```

Dobardan

2.6. Ispis vrijednosti

Ispisivanje vrijednosti varijable u najjednostavnijem obliku izgleda ovako:

```
<?php
  $broj = 5;
  echo $broj;
?>
```

5

Ukoliko se uz varijablu želi ispisati neki tekst, može ga se spojiti s varijablom pomoću operatora spajanja.

```
<?php
  $broj = 5;
  echo "Traženi broj je " . $broj .
      ". Hvala na sudjelovanju.";
?>
```

Traženi broj je 5. Hvala na sudjelovanju.

U jeziku PHP varijabla se može direktno napisati unutar navodnika, a rezultat će biti isti kao da je upotrijebljen operator spajanja.

```
<?php
 $broj = 5;
 echo "Traženi broj je $broj. Hvala na sudjelovanju.";
?>
```

Traženi broj je 5. Hvala na sudjelovanju.

Rezultat je isti kao i u prethodnom primjeru.

No, ako se varijabla napiše unutar jednostrukih navodnika, ispisat će se doslovni sadržaj stringa, dakle neće biti ispisana vrijednost varijable već njeno ime.

```
<?php
 echo 'Traženi broj je $broj. Hvala na sudjelovanju.';
```

```
?>
```

Traženi broj je $broj. Hvala na sudjelovanju.

Ako se unutar dvostrukih navodnika žele spojiti dvije varijable, to se može učiniti ovako:

```
<?php
  $broj1 = 10;
  $broj2 = 20;
  echo "Spojeni brojevi:{$broj1}{$broj2}";
?>
```

Spojeni brojevi: 1020

Unutar dvostrukih navodnika moguće je ispisati i razne specijalne znakove, kao i same znakove **"** ili **$**, navođenjem posebnog niza znakova. Pregled specijalnih znakova dan je u tablici:

Niz znakova	Značenje
\n	znak za prelazak u novi red (*line feed*)
\r	dio kombinacije za prelazak u novi red (*carriage return*)
\t	tabulator
\$	ispisuje znak $
\"	ispisuje znak "
\\	ispisuje znak \

Neki sustavi (*UNIX, Linux*) za prelazak u novi red rabe samo znak *line feed* (\n), dok je na nekima (*Windows*) potrebna kobinacija znakova *carriage return* i *line feed* i *(*\r\n). No, i kod većine Windows aplikacija dovoljan je znak \n za prelazak u novi red.

Ovako bi izgledao niz znakova koji se želi prelomiti u dva reda:

```
"Dobar dan.\nHvala što ste došli."
```

Dobar dan.
Hvala što ste došli.

Znakovi \n, \r i \t imaju ulogu samo prilikom spremanja niza znakova u datoteku (ili ispisa niza znakova unutar naredbenog retka, što se također može raditi iz PHP-a).

Za ispis novog retka u HTML-u, potrebno je ispisati oznaku za novi red (<*br /*>).

Vježba 2.1 – Rad s varijablama

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *index.php*.

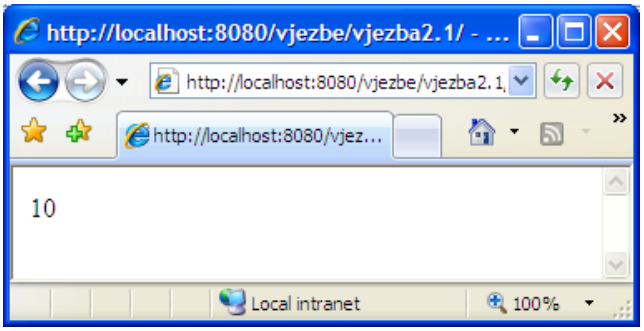
2. U datoteku napišite naredbe kojima ćete varijabli *$a* dodijeliti vrijednost 10 i ispisati vrijednost:

```
<?php

 $a = 10;
 echo $a;

?>
```

Datoteku spremite u mapu "…\D350\vjezbe\vjezba2.1".

3. Pokrenite servis *EasyPHP*, ako nije već pokrenut.

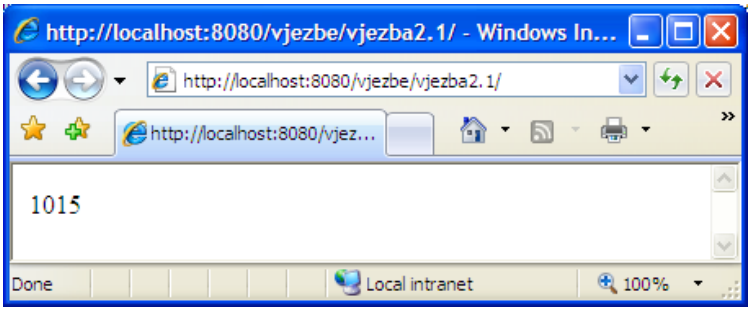
4. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba2.1/* . Uočite da je ispisana vrijednost 10.



5. U datoteku dodajte naredbe kojima ćete varijabli *$a* dodijeliti vrijednost 5 i ispisati vrijednost (masno otisnute naredbe):

```
<?php

 $a = 10;
 echo $a;

 $b = 15;
 echo $b;
?>
```

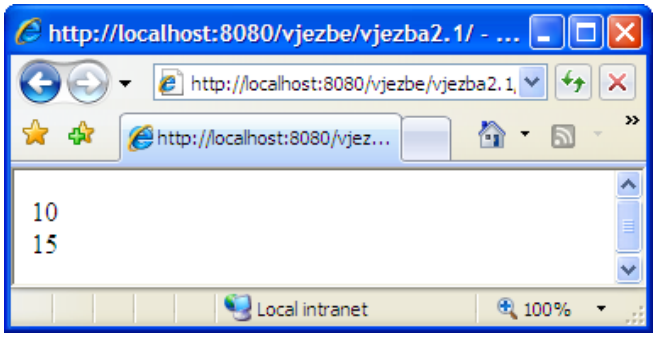
Spremite datoteku i osvježite stranicu u web pregledniku (klikom na dugme  ili kombinacijom tipki Ctrl + F5) .



6. Odvojite vrijednosti 10 i 15 pomoću naredbe koja će ispisati HTML oznaku za novi red:

```
<?php

 $a = 10;
 echo $a;

 echo "<br />";
 $b = 15;
 echo $b;
?>
```

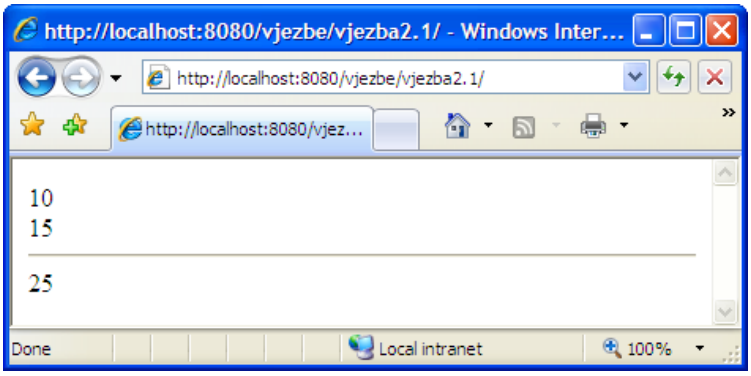
Spremite datoteku i osvježite stranicu u web pregledniku.



7. Na kraj datoteke dodajte naredbu koja će izračunati varijablu *$c* kao zbroj vrijednosti varijabli $a i $b. Prije ispisa rezultata ispišite HTML oznaku za vodoravnu crtu (<hr />).

```
 echo $b;

 echo "<hr />";
 $c = $a + $b;
 echo $c;
?>
```

Spremite datoteku i osvježite stranicu u web pregledniku.

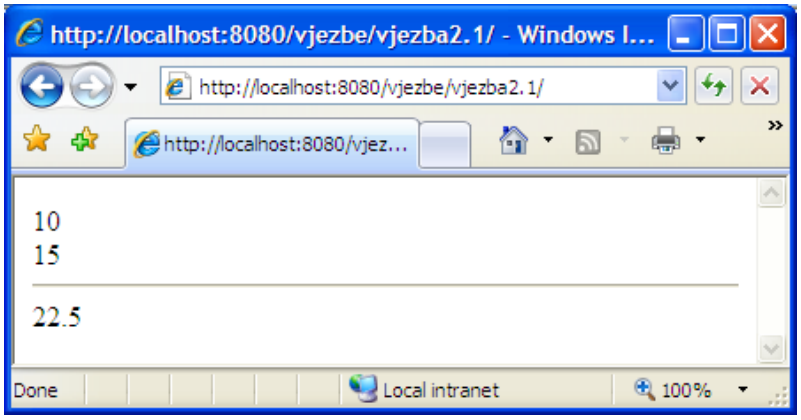


Nakon vodoravne crte ispisan je zbroj vrijednosti 10 i 15.

8. Naredbu koja računa vrijednost varijable $c promijenite tako da se sada vrijednost varijable $c računa kao rezultat izraza 3a – b / 2.

```
 echo "<hr />";
 $c = 3*$a - $b / 2;
 echo $c;
?>
```

Spremite datoteku i osvježite stranicu u web pregledniku.

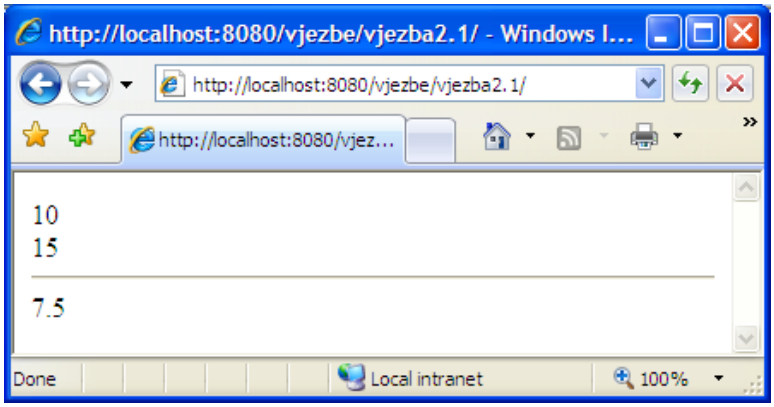


Dobiveni rezultat je 22.5

9. Izmijenite naredbu koja računa vrijednost varijable $c. Dodajte zagrade tako da sada izraz koji se računa bude (3a – b) / 2.

```
 echo "<hr />";
 $c = (3*$a - $b) / 2;
 echo $c;
?>
```

Spremite datoteku i osvježite stranicu u web pregledniku.



Dobivena vrijednost sada je 7.5 zato što je promijenjen redoslijed izvršavanja aritmetičkih operacija.

10. Obrišite sve *echo* naredbe.

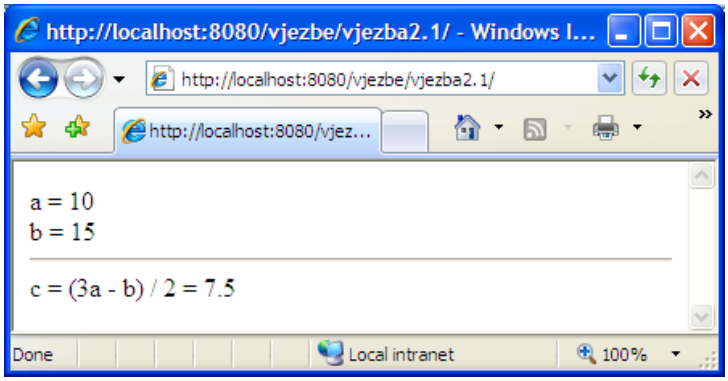
11. Iza preostalih naredbi dodajte naredbe koje će varijablama *$tekst1*, *$tekst2* i *$tekst3* pridružiti tekstualne opise varijabli *$a*, *$b* i *$c*:

```
  $c = (3*$a - $b) / 2;

  $tekst1 = "a = " . $a;
  $tekst2 = "b = " . $b;
  $tekst3 = "c = (3a - b)/2 = " . $c;
?>
```

12. Dodajte naredbu koja će ispisati kombinaciju varijabli *$tekst1*, *$tekst2* i *$tekst3*, ali uz razdvajanje HTML oznakama za novi red i vodoravnom crtom:

```
  $tekst3 = "c = (3a - b) / 2 = " . $c;

  echo $tekst1 . "<br />" . $tekst2 . "<hr />" . $tekst3;
?>
```

Spremite datoteku i osvježite stranicu u web pregledniku.



13. Promijenite posljednje 4 naredbe tako da se ne koristite operatorom **.** za spajanje znakovnih nizova, već da varijable ubacite unutar navodnika:

```
  $tekst1 = "a = $a";
  $tekst2 = "b = $b";
  $tekst3 = "c = (3a - b) / 2 = $c";

  echo "$tekst1<br />$tekst2<hr />$tekst3";
?>
```

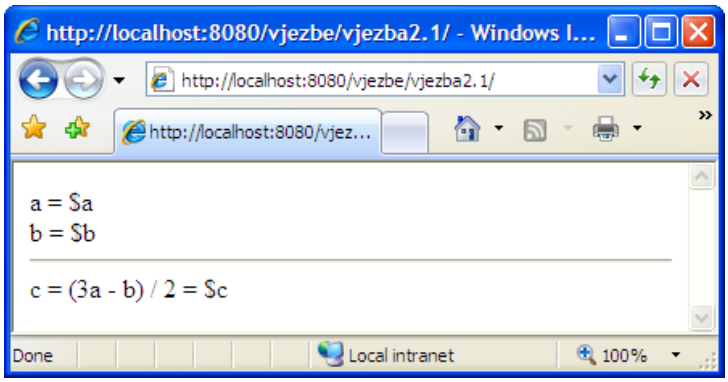
Spremite datoteku i osvježite stranicu u web pregledniku. Rezultat je isti.

Budući da se koriste dvostruki navodnici, moguće je ubaciti varijable unutar navodnika i na taj način izvršiti spajanje znakovnih nizova.

14. Promijenite sve navodnike u ovim trima naredbama u jednostruke:

```
  $tekst1 = 'a = $a';
  $tekst2 = 'b = $b';
  $tekst3 = 'c = (3a - b) / 2 = $c';
?>
```

Spremite datoteku i osvježite stranicu u web pregledniku.



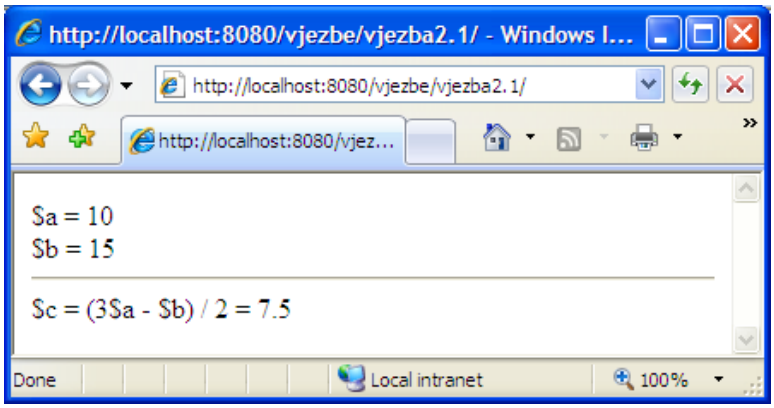
Dobiveni rezultat pokazuje da ako se ime varijable upiše unutar jednostrukih navodnika, niz znakova koji nastaje ne sadrži vrijednost varijable (kao u slučaju dvostrukih navodnika), već samo znakove koji tvore njeno ime.

15. Vratite navodnike natrag u dvostruke.

16. Da bismo dobili ispisano puno ime varijable (zajedno sa znakom $), u tekstu koji opisuje varijablu dodajte znak ***$*** pomoću specijalnog niza znakova ("*\$*").

```
  $tekst1 = "\$a = $a";
  $tekst2 = "\$b = $b";
  $tekst3 = "\$c = (3\$a - \$b) / 2 = $c";
?>
```

Spremite datoteku i osvježite stranicu u web pregledniku.



U ovom je poglavlju obrađeno:

- varijable i tipovi podataka u jeziku PHP
- pridruživanje vrijednosti varijablama
- primjena aritmetičkih operatora
- spajanje znakovnih nizova
- mogućnosti ispisa vrijednosti varijable pomoću naredbe *echo*

3. Operatori usporedbe, logički operatori i uvjetne strukture

3.1. Operatori usporedbe

Operatori usporedbe služe, kao što im ime kaže, za uspoređivanje vrijednosti. Rezultat koji vraća operacija usporedbe logička je vrijednost *TRUE* ili *FALSE*, ovisno o istinitosti usporedbe.

Operator	Značenje	Primjer	Rezultat
==	jednako	`$a == $b`	*TRUE* ako je *$a* jednako *$b*
===	identično	`$a === $b`	*TRUE* ako je *$a* jednako *$b* i ako su *$a* i *$b* istog tipa podataka
!=	nije jednako	`$a != $b`	*TRUE* ako *$a* nije jednako *$b*
<>		`$a <> $b`	
!==	nije identično	`$a !== $b`	*TRUE* ako *$a* nije jednako *$b* ili ako nisu istog tipa podataka
<	manje od	`$a < $b`	*TRUE* ako je *$a* strogo manje od *$b*
>	više od	`$a > $b`	*TRUE* ako je *$a* strogo više od *$b*
<=	manje ili jednako	`$a <= $b`	*TRUE* ako je *$a* manje ili jednako *$b*
>=	više ili jednako	`$a >= $b`	*TRUE* ako je *$a* više ili jednako *$b*

Operator jednakosti (==) ne treba miješati s operatorom pridruživanja (=). Treba paziti da se prilikom uspoređivanja dviju vrijednosti greškom ne upotrijebi operator = umjesto operatora == (što je česta pogreška). PHP prevoditelj to neće prepoznati kao grešku, a dobiveni rezultat neće biti točan.

PHP dozvoljava usporedbu vrijednosti različitih tipova podataka. Ako se uspoređuje brojčana vrijednost s tekstualnom, tekstualna vrijednost se pretvara u ekvivalentnu brojčanu i vrši se usporedba dvije brojčane vrijednosti. Ako je pretvorba nemoguća, odnosno varijable nisu ekvivalentnih vrijednosti, rezultat usporedbe je *FALSE*.

Operator identičnosti (**===**) služi za provjeru jesu li dvije varijable, osim što sadržavaju ekvivalentne vrijednosti, i istog tipa podataka.

Sljedeći primjer prikazuje uspoređivanje vrijednosti u jeziku PHP:

```
<?php
  $a = 1;
  $b = 2;
  $a_tekst = "1";

  echo $a == $b;          // FALSE (ispisuje "")
  echo $a < $b;           // TRUE (ispisuje "1")
  echo $a == $a_tekst;    // TRUE (ispisuje "1")
  echo $a === $a_tekst;   // FALSE (ispisuje "")
  echo $a_tekst < $b;     // TRUE (ispisuje "1")
?>
```

U ovom bi primjeru naredba *echo* za svaku vrijednost *TRUE* ispisala "1", a za svaku vrijednost *FALSE* ispisala bi "", tj. prazan niz. Razlog tome je što naredba *echo* pretvara dobivene argumente u tekstualni tip podataka. Logički tip podatka (*boolean*) se pretvara u tekstualni (*string*) po gornjem principu.

Pored svake naredbe, u kojoj se vrši uspoređivanje, u komentarima je dan rezultat usporedbe.

Rezultat prve od tih naredbi je *FALSE*, a druge *TRUE* jer je vrijednost *$a* manja od vrijednosti *$b*. Treća naredba daje rezultat *TRUE* jer su vrijednosti *$a* i *$a_tekst* ekvivalentne, no četvrta naredba, koja se koristi operatorom ===, daje rezultat *FALSE* zato što su te dvije varijable različitih tipova podataka. Posljednja naredba daje rezultat *TRUE* jer je pretvorena vrijednost varijable *$a_tekst* manja od vrijednosti varijable *$b*.

3.2. Logički operatori

Uz aritmetičke operacije i operacije uspoređivanja, u programskim jezicima postoje i logičke operacije. Operandi, kao i rezultati u logičkim operacijama, su logičke vrijednosti (*TRUE* i *FALSE*). Za pisanje logičkih izraza koriste se logički operatori čiji je pregled dan u sljedećoj tablici:

Operator	Značenje	Prioritet	Primjer	Rezultat
`!`	negacija	1	`!$a`	*TRUE* ako je *$a FALSE*
`&&`	logičko i	2	`$a && $b`	*TRUE* ako su obje vrijednosti *TRUE*
`and`		3	`$a and $b`	
`||`	logičko ili	4	`$a || $b`	*TRUE* ako je bar jedna od vrijednosti *TRUE*
`or`		6	`$a or $b`	
`xor`	ekskluzivno ili	5	`$a xor $b`	*TRUE* ako je samo jedna vrijednost *TRUE*

Operatori *&&* i *and* su međusobno jednaki, baš kao i operatori || i *or*, samo što prvi ima viši prioritet pri redoslijedu izračunavanja logičkih izraza.

Slijedi jednostavan primjer upotrebe logičkih operatora:

```
<?php
  $a = TRUE;
  $b = FALSE;

  echo !$a;          // FALSE (ispisuje "")
  echo $a and $b;    // FALSE (ispisuje "")
  echo $a or $b;     // TRUE (ispisuje "1")
  echo $a xor $b;    // TRUE (ispisuje "1")
?>
```

Prva u posljednjem bloku naredbi dat će rezultat FALSE, jer se negira vrijednost varijable *$a*. Druga će naredba dati rezultat *FALSE* jer je jedna od vrijednosti jednaka *FALSE* (*$b*). Treća naredba dat će rezultat *TRUE* jer je jedna od vrijednosti (*$a*) jednaka *TRUE*. Četvrta naredba također će dati rezultat TRUE, i to zato što je samo vrijednost (*$a*) jednaka *TRUE*, a druga je vrijednost *($b*) *FALSE*.

Kod slaganja složenijih logičkih izraza, također se, kao i kod aritmetičkih izraza, mogu koristiti zagrade za promjenu redoslijeda izvršavanja, ali i radi bolje čitljivosti izraza.

```
<?php
  $c = TRUE;
  echo $a and ($b or !$c);   // FALSE (ispisuje "")
?>
```

Uz vrijednosti varijabli $a i $b kao u prethodnim primjerima, druga naredba u gornjem primjeru dala bi rezultat *FALSE*.

3.3. Jednostavne uvjetne strukture

U svim programskim jezicima, pa tako i u skriptnom jeziku PHP, postoje uvjetne strukture koje se koriste za donošenje odluke o daljnjem tijeku izvođenja programa. Ovisno o tome je li neki logički uvjet ispunjen (odnosno je li njegova vrijednost *TRUE* ili *FALSE*), obavit će se ili se neće obaviti neke naredbe.

If struktura

Jednostavna uvjetna struktura ima ovakav oblik:

```
  if (uvjet)
  {
    naredba1;
    naredba2;
  }
```

Nakon ključne riječi *if* u zagradama se navodi uvjet (može biti logička vrijednost, varijabla ili logički izraz). Unutar vitičastih zagrada navode se naredbe koje će biti izvršene ako je uvjet ispunjen.

U sljedećem primjeru rečenica "a je veće od b" ispisat će se samo ako je vrijednost *$a* doista veća od vrijednosti *$b*.

Ako vrijednost ili varijabla unutar zagrada nije logičkog tipa, izvršit će se njena pretvorba u logičku vrijednost.

```
<?php
  if ($a > $b)
  {
    echo "a je veće od b.";
  }
?>
```

Ako se unutar vitičastih zagrada nalazi samo jedna naredba, zagrade se mogu i izostaviti:

```
<?php
  if ($a > $b)
    echo "a je veće od b.";
?>
```

If – else struktura

Dodavanjem ključne riječi *else* mogu se odrediti naredbe koje će biti izvršene ako uvjet nije ispunjen:

```
  if (uvjet)
  {
    naredba1;
  }
  else
  {
    naredba2;
  }
```

U sljedećem primjeru ispisat će se, ako je vrijednost *$a* veća od vrijednosti *$b*, jedna poruka, a u suprotnom (ako je manja ili jednaka) ispisat će se druga.

```
<?php
  if ($a > $b)
  {
    echo "a je veće od b.";
  }
  else
  {
    echo "a je manje ili jednako b.";
  }
?>
```

3.4. Složene uvjetne strukture

U složenijim situacijama potrebno je ispitati više uvjeta prije nego što se donese odluka.

If – elseif struktura

Jednostavna uvjetna struktura može se proširiti dodavanjem novog uvjeta, uz korištenje ključne riječi *elseif*.

```
  if (uvjet1)
  {
    naredba1;
  }
  elseif (uvjet2)
  {
    naredba2;
  }
  else
  {
    naredba3;
  }
```

U slučaju da je prvi uvjet ispunjen, izvršava se prva naredba. Ako prvi uvjet nije ispunjen, provjerit će se je li drugi uvjet ispunjen, i ako jest, izvršava se druga naredba. Ako nijedan ni drugi uvjet nije ispunjen, izvršava se treća naredba.

Sljedeći primjer će ispisati odgovarajuću poruku za svaki od tri slučajeva – ako je *$a* veće od *$b*, ako je manje i na kraju, ako su vrijednosti jednake.

```
<?php
  if ($a > $b)
  {
    echo "a je veće od b.";
  }
  elseif ($a < $b)
  {
    echo "a je manje od b.";
  }
  else
  {
    echo "a je jednako b.";
  }
?>
```

Moguće je dodati proizvoljan broj *elseif* blokova:

```
<?php
  if ($a == 0)
  {
    echo "a je jednako 0.";
  }
  elseif ($a == 1)
  {
    echo "a je jednako 1.";
  }
  elseif ($a == 2)
  {
    echo "a je jednako 2.";
  }
  else
  {
    echo "a nije jednako 0, 1 niti 2.";
  }
?>
```

Važno je zapamtiti da će se izvršiti naredbe uz prvi navedeni uvjet koji je istinit, a ako nijedan uvjet nije istinit, izvršavaju se naredbe unutar *else* bloka.

U *if-elseif* strukturi *else* blok se može i izostaviti.

Switch struktura

If – elseif struktura u kojoj svi uvjeti provjeravaju vrijednost neke varijable ili izraza može se napisati na jednostavniji način, korištenjem ključne riječi *switch*.

Bitno je ne izostaviti naredbu *break* na kraju *case* bloka. U suprotnom će se izvršiti i sve naredbe navedene u blokovima ispod bloka čiji je uvjet ispunjen.

```
<?php
  switch (izraz)
  {
    case vrijednost1:
      naredba1;
      naredba2;
      break;
    case vrijednost2:
      naredba3;
      naredba4;
      break;
    default:
      naredba5;
      naredba6;
  }
?>
```

Ovisno o vrijednosti izraza navedenog unutar zagrada poslije ključne riječi *switch*, izvršit će se naredbe koje se nalaze unutar odgovarajućeg *case* bloka (onoga u kojem je navedena točna vrijednost). Nakon što se pronađe točan *case* blok, izvršavaju se sve naredbe dok se ne naiđe na naredbu *break*. Ukoliko odgovarajuća vrijednost nije pronađena, izvršavaju se naredbe navedene poslije ključne riječi *default*.

Primjer s provjerom vrijednosti varijable $a izgledao bi, napisan pomoću *switch* strukture, ovako:

```
<?php
  switch ($a)
  {
    case 0:
      echo "a je jednako 0.";
      break;
    case 1:
      echo "a je jednako 1.";
      break;
    case 2:
      echo "a je jednako 2.";
      break;
    default:
      echo "a nije jednako 0, 1 niti 2.";
  }
?>
```

Vrijednosti koje se ispituju mogu se grupirati zajedno. U ovom primjeru će se izvršiti ista naredba ako je $a jednako 0 i ako je *$a* = 1:

```
<?php
  switch ($a)
  {
    case 0:
    case 1:
      echo "a je jednako 0 ili 1.";
      break;
    case 2:
      echo "a je jednako 2.";
      break;
    default:
      echo "a nije jednako 0, 1 niti 2.";
  }
?>
```

3.5. Ugnježđivanje uvjetnih struktura

Kod ispitivanja složenih uvjeta, moguće je ugnijezditi jednu uvjetnu strukturu unutar druge. Slijedi primjer takvog ugnježđivanja:

```
<?php
  $pada_kisa = TRUE;
  $trava_raste = FALSE;

  if ($pada_kisa)
  {
    if ($trava_raste)
    {
      echo "Pada kiša, trava raste.";
    }
```

```
    else
    {
      echo "Pada kiša, ali trava ne raste!?";
    }
  }
  else
  {
    echo "Ne pada kiša.";
  }
?>
```

U ovom je primjeru jedna provjera uvjeta ugnježđena unutar druge. Samo kad je ispunjen prvi uvjet (ako je vrijednost *$pada_kisa* jednaka *TRUE*), izvršit će se provjera drugog uvjeta.

Više ugnježđenih struktura može se uvijek napisati kao jedna *if-elseif* struktura, ali će se pritom često izgubiti na čitljivosti. Isti primjer, napisan pomoću jedne strukture, izgledat će ovako:

```
<?php
  if ($pada_kisa && $trava_raste)
  {
    echo "Pada kiša, trava raste.";
  }
  else if ($pada_kisa && !$trava_raste)
  {
    echo "Pada kiša, ali trava ne raste!?";
  }
  else
  {
    echo "Ne pada kiša.";
  }
?>
```

Vježba 3.1 – Uvjetne strukture

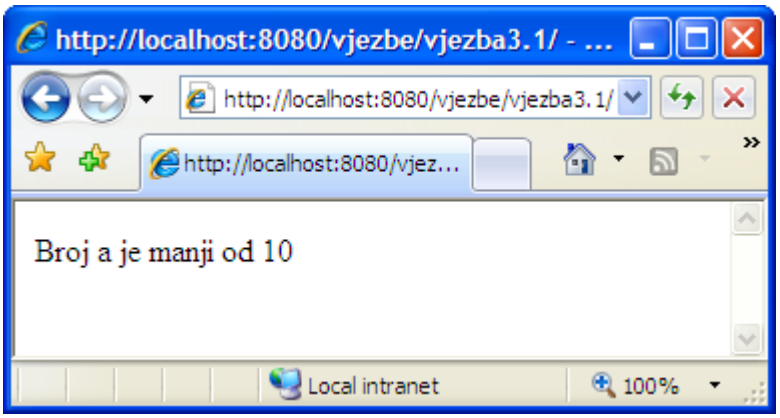
1. Pokrenite program *PHP Designer* i stvorite novu datoteku *index.php*.
2. U datoteku upišite sljedeće naredbe:

```
<?php
  $a = 1;

  if ($a < 10)
  {
    echo "Broj a je manji od 10";
  }
?>
```

   Varijabli *$a* pridružuje se vrijednost 1, a nakon toga se pomoću *if* strukture provjerava je li vrijednost varijable manja od 10.

   Datoteku spremite u mapu "C:\www\D350\vjezbe\vjezba3.1".

3. Pokrenite servis *EasyPHP*, ako nije već pokrenut.
4. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba3.1/* . Očekivano, ispisuje se poruka da je broj a manji od 10.

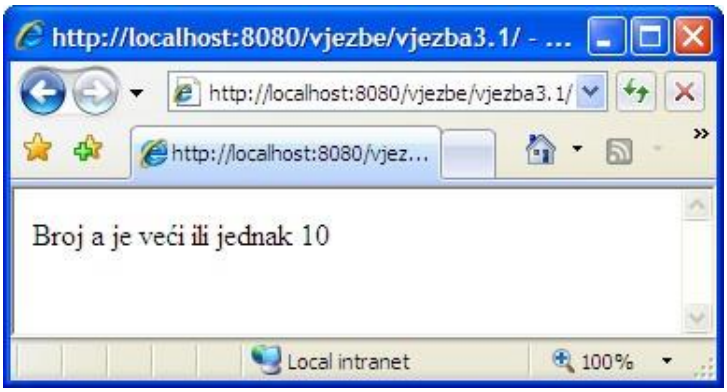


5. Promijenite *if* strukturu tako da joj dodate i *else* blok koji će ispisivati poruku kad *$a* ne bude manje od 10. Promijenite vrijednost varijable $a, tako da ona sad bude 11.

```
<?php
  $a = 11;

  if ($a < 10)
  {
    echo "Broj a je manji od 10";
  }
  else
  {
    echo "Broj a je veći ili jednak 10";
  }
?>
```

6.

7. Osvježite stranicu u web pregledniku. Kako je vrijednost broja a promijenjena, ispisat će se druga poruka:

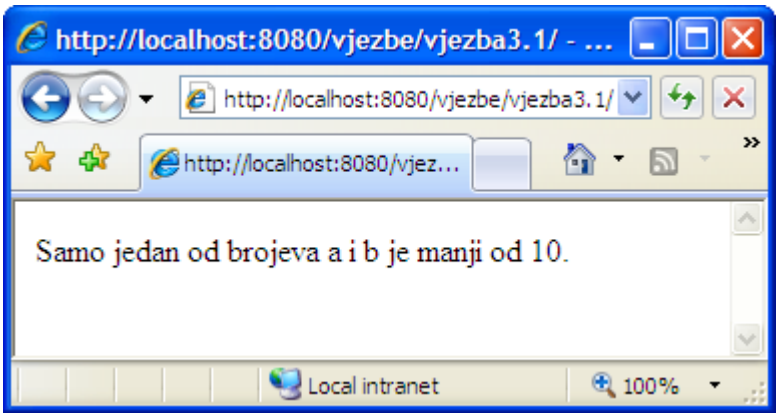


8. Dodajte varijablu *$b* s vrijednošću 5. Promijenite uvjetnu strukturu tako da se sada ispituje i vrijednost varijable *$b*, i da se doda jedan *elseif* blok:

```
<?php
  $a = 11;
  $b = 5;

  if ($a < 10 and $b < 10)
  {
    echo "Brojevi a i b su manji od 10.";
  }
  elseif ($a < 10 or $b < 10)
  {
    echo "Samo jedan od brojeva a i b je manji od 10.";
  }
  else
  {
    echo "Nijedan od brojeva a i b nije manji od 10.";
  }
?>
```

Uz korištenje logičkih operatora *and* i or, u uvjetima se ispituju vrijednosti varijabli *$a* i *$b*. Prvi uvjet će biti ispunjen ako su i *$a* i *$b* manji od 10. Ako taj uvjet nije ispunjen, ispitivat će se sljedeći uvjet, koji će biti ispunjen ako je bar jedan od brojeva manji od 10. Ako ni jedan ni drugi uvjet nije ispunjen, izvršit će se naredba u *else* bloku.

9. Osvježite stranicu u web pregledniku. Budući da je $a jednako 11, a $b jednako 5, ispisat će se poruka da je samo jedan od brojeva manji od 10.



U ovom je poglavlju obrađeno:

- operatori usporedbe i uspoređivanje varijabli
- logički operatori i logički izrazi
- jednostavne i složene uvjetne strukture
- ugnježđivanje uvjetnih struktura

4. Polja

Za grupiranje više vrijednosti, u programskim jezicima koristi se koncept polja. Polje je varijabla koja sadrži niz vrijednosti, a svakoj vrijednosti (članu polja) pridružen je ključ. Pojedinom članu polja pristupa se pomoću ključa, koji može biti broj ili niz znakova.

Vrijednosti koje polje sadržava mogu biti bilo kojeg tipa podatka podržanog u jeziku PHP.

Polja mogu biti jednodimenzionalna, ali i dvodimenzionalna i višedimenzionalna.

4.1. Polja s brojčanim ključem (indeksom)

Brojčani ključ je obično redni broj člana u polju, i tada se naziva **indeksom**. Uobičajeno je da brojanje članova počinje od nule, tako da je indeks prvog člana polja 0, drugog 1, trećeg 2 i tako dalje.

Tablica 4.1 prikazuje primjer polja koje sadrži imena gradova.

Indeks	Vrijednost
0	"Zagreb"
1	"Split"
2	"Rijeka"

Za stvaranje polja koristi se ključna riječ *array*:

```
<?php
  $gradovi = array("Zagreb","Split","Rijeka");
?>
```

Za pristup članu polja koristi se njegov indeks, napisan unutar uglatih zagrada. Sljedeći primjer ispisat će vrijednost člana polja s indeksom 2:

```
<?php
  echo $gradovi[2];
?>
```

```
Rijeka
```

Osim prilikom stvaranja polja, članovima polja je moguće i pojedinačno pridruživati vrijednosti:

```
<?php
  $gradovi[0] = "Zagreb";
  $gradovi[1] = "Split";
  $gradovi[2] = "Rijeka";
?>
```

Polje ne mora biti prethodno stvoreno pomoću ključne riječi *array*. U tom slučaju, bit će stvoreno nakon što se jednom članu polja pridruži vrijednost.

Već stvoreno polje nema fiksnu veličinu, odnosno moguće mu je kasnije nadodati proizvoljan broj članova.

```
<?php
  $gradovi[3] = "Osijek";
?>
```

No, pristup vrijednosti članu polja koji nije definiran uzrokovat će pogrešku.

4.2. Polja sa znakovnim ključem

Osim broja, ključ polja može biti i znakovni niz. Ideja je da takav ključ bude naziv koji je smisleno povezan s vrijednošću člana polja. Polja sa znakovnim ključem nazivaju se stoga i **asocijativna polja**.

Za polje koje treba sadržavati poštanske brojeve gradova, ključ polja može biti naziv grada.(Tablica 4.2)

Ključ	Vrijednost
"Zagreb"	10 000
"Split"	21 000
"Rijeka"	51 000

Pri stvaranju polja sa znakovnim ključem, za svaki član potrebno je navesti i ključ i vrijednost:

```
<?php
  $post_br = array ("Zagreb" => 10000,
                    "Split" => 21000,
                    "Rijeka" => 51000);
?>
```

Za pristup članu polja koristi se njegov ključ:

```
<?php
  echo $post_br["Rijeka"];
?>
```

```
51000
```

Pridruživanje vrijednosti članu polja također se obavlja preko njegovog ključa:

```
<?php
  $post_br["Osijek"] = 31000;
?>
```

4.3. Dvodimenzionalna polja

Dok se jednodimenzionalno polje može predočiti kao niz vrijednosti, dvodimenzionalno polje može se predočiti kao tablica. Član dvodimenzionalnog polja određen je s dva ključa – po jedan za svaku dimenziju.

Primjer za dvodimenzionalno polje bi moglo biti trenutno stanje u igri "križić-kružić". Budući da je riječ o polju dimenzija 3x3, vrijednosti za oba indeksa (okomiti i vodoravni) su od 0 do 2.

	0	**1**	**2**
0	O	O	
1	O	X	O
2	X	O	X

Tablica 4.2: Polje u koje se sprema stanje u igri "križić-kružić" (u zaglavlju i u prvom stupcu su indeksi polja)

Dvodimenzionalno polje u koje se sprema gornje stanje u igri bit će stvoreno na ovaj način:

```
<?php
  $igra = array( array ("O", "O", ""),
                 array ("O", "X", "O"),
                 array ("X", "O", "X") );
?>
```

Dvodimenzionalno polje stvara se tako da se definira polje koje kao članove ima jednodimenzionalna polja.

Za pristup vrijednosti člana dvodimenzionalnog polja potrebno je navesti oba indeksa. Prvi indeks označava redak, a drugi stupac u kojem se nalazi član. Sljedeća naredba će ispisati vrijednost prvog člana u drugom stupcu:

```
<?php
  echo $igra[1][0];
?>
```

```
O
```

Na ovaj način bi se popunio preostali član polja (koji nije ni križić ni kružić):

```
<?php
  $igra[0][2] = "X";
?>
```

Ako se retke, umjesto brojevima 0,1 i 2, želi označiti sa slovima A, B i C, potrebno je stvoriti polje na sljedeći način:

```
<?php
  $igra = array( "A" => array ("O", "O", ""),
                 "B" => array ("O", "X", "O"),
                 "C" => array ("X", "O", "X") );
?>
```

Sad bi postavljanje završnog križića izgledalo ovako:

```
<?php
  $igra["A"][2] = "X";
?>
```

Moguće je definirati polje s neograničenim brojem dimenzija, iako polja s više od 3 dimenzije nemaju neku praktičnu primjenu. Naredba kojom se stvara trodimenzionalno polje slična je kao i naredba za stvaranje dvodimenzionalnog polja, samo se sada definira polje čiji je svaki član dvodimenzionalno polje. Za pristup članu trodimenzionalnog polja bit će potrebna tri indeksa.

Vježba 4.1 – Polje s brojčanim ključem

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *index.php*.
2. U datoteku upišite oznake za PHP kôd i unutar njih sljedeću naredbu koja će stvoriti polje *ocjene*. Ono će služiti za spremanje učenikovih ocjena iz tri predmeta:

```
<?php
  $ocjene = array(4,5,4);
?>
```

3. U datoteku dodajte naredbu koja će izračunati prosjek ocjena:

```
  $prosjek = ($ocjene[0] + $ocjene[1] + $ocjene[2]) / 3;
?>
```

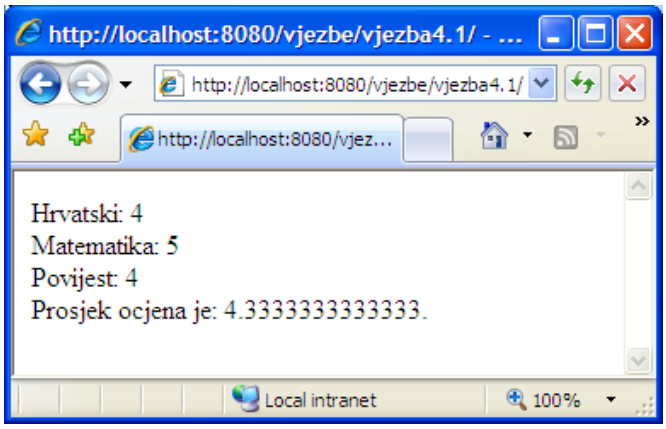
   Prosjek ocjena računamo tako da zbrojimo sve ocjene iz polja i zbroj podijelimo s tri (ukupnim brojem ocjena).

4. U datoteku upišite sljedeće naredbe koje će ispisati pojedine ocjene i njihov prosjek:

```
  $prosjek = ($ocjene[0] + $ocjene[1] + $ocjene[2]) / 3;

  echo "Hrvatski: $ocjene[0]<br />";
  echo "Matematika: $ocjene[1]<br />";
  echo "Povijest: $ocjene[2]<br />";
  echo "Prosjek ocjena je: $prosjek.";
?>
```

   Datoteku spremite u mapu "..\D350\vjezbe\vjezba4.1".

5. Pokrenite servis *EasyPHP*, ako nije već pokrenut.
6. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba4.1/*



Vježba 4.2 – Polje sa znakovnim ključem

Ova vježba se nastavlja na vježbu 4.1.

1. U programu *PHP Designer* otvorite datoteku *index.php* u mapi "..\D350\vjezbe\vjezba4.2".

2. Promijenite naredbu koja stvara polje tako da stvoreno polje ima znakovni ključ koji će biti naziv predmeta.

```
<?php
  $ocjene = array("Hrvatski" => 4,
                  "Matematika" => 5,
                  "Povijest" => 4);
?>
```

3. Naredbu koja računa prosjek ocjena potrebno je promijeniti tako da se umjesto indeksa, za dohvat ocjene koristi naziv predmeta:

```
  $prosjek = ( $ocjene["Hrvatski"] + $ocjene["Matematika"] +
               $ocjene["Povijest"] ) / 3;
?>
```

4. Naredbe koje ispisuju pojedine ocjene također je potrebno promijeniti. No, budući da se koristi polje sa znakovnim ključem, da bi se ispravno ispisao član polja koji je ubačen u znakovni niz s dvostrukim navodnicima, potrebno ga je staviti unutar vitičastih zagrada.

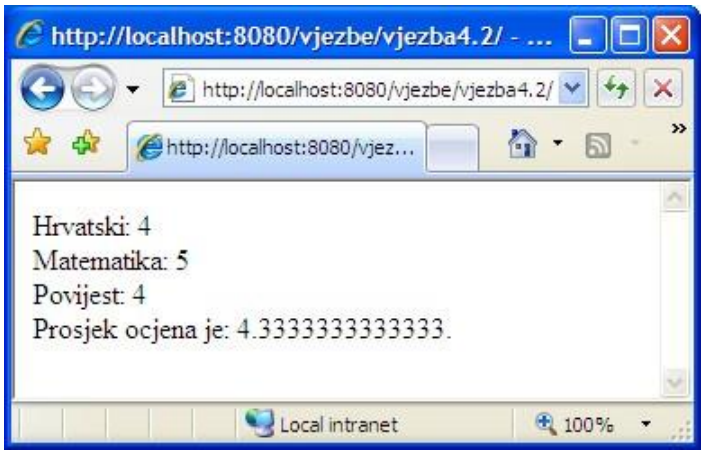
Član polja sa znakovnim ključem, ubačen u niz s dvostrukim navodnicima, neće biti ispisan ispravno ako ne bude stavljen unutar vitičastih zagrada. Ni korištenje jednostrukih navodnika za pisanje ključa polja neće biti dovoljno da član polja bude prepoznat kao varijabla.

```
  echo "Hrvatski: {$ocjene["Hrvatski"]}<br />";
  echo "Matematika: {$ocjene["Matematika"]}<br />";
  echo "Povijest: {$ocjene["Povijest"]}<br />";
  echo "Prosjek ocjena je: $prosjek.";
?>
```

   Spremite datoteku.

5. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

6. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba4.2/*



Rezultat će biti isti kao i u prethodnoj vježbi. No, sada nije bilo potrebno pamtiti kojem predmetu pripada koji indeks, već se ocjeni moglo pristupiti preko naziva predmeta.

Vježba 4.3 – Dvodimenzionalno polje

Ova vježba se nastavlja na vježbu 4.2.

1. U programu *PHP Designer* stvorite novu datoteku *index.php*.
2. U datoteku upišite oznake za PHP kôd i unutar njih naredbu koja će stvoriti dvodimenzionalno polje *ocjene*. Polje će služiti za spremanje ocjena nekoliko učenika:

```
<?php
  $ocjene = array ( "Ivica" => array("Hrvatski" => 4,
                                     "Matematika" => 5,
                                     "Povijest" => 4 ),
                    "Tomica" => array("Hrvatski" => 5,
                                      "Matematika" => 5,
                                      "Povijest" => 4 ),
                    "Perica" => array("Hrvatski" => 5,
                                      "Matematika" => 3,
                                      "Povijest" => 4 ) );
?>
```

3. Za pristupanje ocjeni sad su potrebna dva ključa – ime učenika i naziv predmeta. Dodajte naredbe koje će izračunati prosjek za svakog pojedinog učenika:

```
  $prosjekIvica = ( $ocjene["Ivica"]["Hrvatski"] +
                    $ocjene["Ivica"]["Matematika"] +
                    $ocjene["Ivica"]["Povijest"] ) / 3;

  $prosjekTomica = ( $ocjene["Tomica"]["Hrvatski"] +
                     $ocjene["Tomica"]["Matematika"] +
                     $ocjene["Tomica"]["Povijest"] ) / 3;

  $prosjekPerica = ( $ocjene["Perica"]["Hrvatski"] +
                     $ocjene["Perica"]["Matematika"] +
                     $ocjene["Perica"]["Povijest"] ) / 3;
?>
```

4. Izvan oznaka za PHP kôd dodajte HTML oznake za tablicu i zaglavlje tablice. Zaglavlje će sadržavati nazive predmeta i tekst "Prosjek".

```
?>
<table border="1" cellpadding="5">
   <tr>
     <th></th>
     <th>Hrvatski</th>
     <th>Matematika</th>
     <th>Povijest</th>
     <th>Prosjek</th>
   </tr>

</table>
```

Ako se unutar oznaka za PHP kôd nalazi smao jedna naredba, nije potrebno pisati znak **;** na kraju naredbe.

5. Poslije zaglavlja, u tablicu dodajte tri retka koji će sadržavati ime učenika, ocjene iz pojedinih predmeta i učenikov prosjek. Sve osim imena učenika bit će ispisano iz PHP-a, pomoću naredbe *echo*, za što je potrebno unutar svake oznake td umetnuti oznake za PHP kôd:

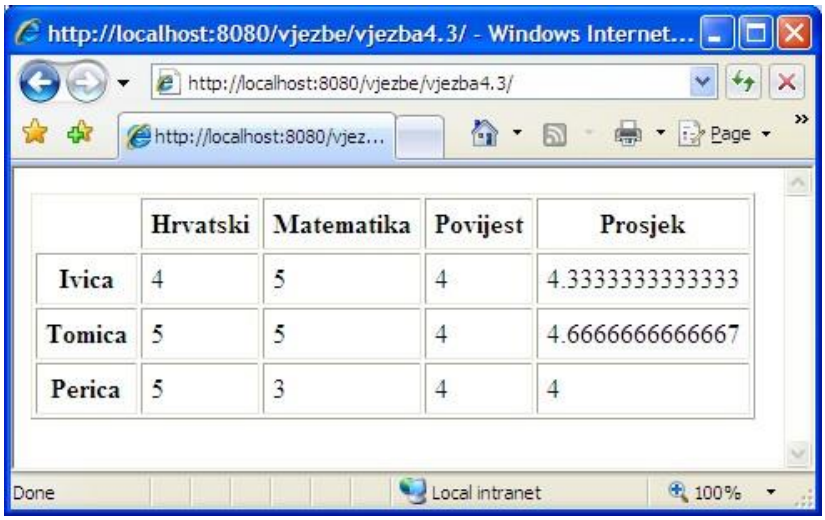
```
   </tr>
   <tr>
     <th>Ivica</th>
     <td><?php echo $ocjene["Ivica"]["Hrvatski"] ?></td>
     <td><?php echo $ocjene["Ivica"]["Matematika"] ?></td>
     <td><?php echo $ocjene["Ivica"]["Povijest"] ?></td>
     <td><?php echo $prosjekIvica ?></td>
   </tr>
   <tr>
     <th>Tomica</th>
     <td><?php echo $ocjene["Tomica"]["Hrvatski"] ?></td>
     <td><?php echo $ocjene["Tomica"]["Matematika"] ?></td>
     <td><?php echo $ocjene["Tomica"]["Povijest"] ?></td>
     <td><?php echo $prosjekTomica ?></td>
   </tr>
   <tr>
     <th>Perica</th>
     <td><?php echo $ocjene["Perica"]["Hrvatski"] ?></td>
     <td><?php echo $ocjene["Perica"]["Matematika"] ?></td>
     <td><?php echo $ocjene["Perica"]["Povijest"] ?></td>
     <td><?php echo $prosjekPerica ?></td>
   </tr>
</table>
```

Datoteku spremite u mapu "..\D350\vjezbe\vjezba4.3".

6. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

7. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba4.3/* .

Konačni rezultat izgledat će ovako:



	Hrvatski	Matematika	Povijest	Prosjek
Ivica	4	5	4	4.3333333333333
Tomica	5	5	4	4.6666666666667
Perica	5	3	4	4

U ovom je poglavlju obrađeno:

- koncept polja
- polja s brojčanim indeksima
- asocijativna polja
- dvodimenzionalna i višedimenzionalna polja

5. Petlje

Petlje su izrazi kojima se ostvaruje ponavljanje određene naredbe ili bloka naredbi. Na taj način nije potrebno više puta napisati naredbe koje se trebaju izvršiti više puta, već ih je dovoljno napisati unutar petlje.

Naredbe će se ponavljati dok je ispunjen određeni logički uvjet.

5.1. Petlja *while*

Najjednostavnija petlja je petlja *while*. Ovako izgleda njen osnovni oblik:

```
while (uvjet)
{
  naredba1;
  naredba2;
}
```

Poslije ključne riječi *while* u zagradama se piše uvjet ponavljanja. Nakon njega dolazi tijelo petlje – unutar vitičastih zagrada se navodi naredba ili naredbe koje će se ponavljati dok god uvjet vrijedi. Kao i kod uvjetnih struktura, moguće je izostaviti vitičaste zagrade ako se radi samo o jednoj naredbi.

Istinitost uvjeta se provjerava svaki put prije nego što se izvrše naredbe iz tijela petlje. Nakon što uvjet prestane vrijediti, izvršavanje programa se nastavlja od prve naredbe koja slijedi nakon petlje. Ako uvjet petlje nije bio ispunjen na početku, naredbe u tijelu petlje neće biti izvršene nijednom.

Istinitost uvjeta se može promijeniti kao rezultat izvršavanja naredbi u tijelu petlje, ili zbog nekog vanjskog uzroka. Sljedeća petlja izvršit će se 10 puta, zato što nakon 10. izvršavanja uvjet petlje više neće vrijediti:

```
<?php
  $i = 0;
  while($i < 10)
  {
    echo $i . " ";
    $i++;
  }
?>
```

```
0 1 2 3 4 5 6 7 8 9
```

U gornjem primjeru je najprije stvorena varijabla *$i*, kojoj je pridružena početna vrijednost 0. Zatim slijedi petlja čiji je uvjet ponavljanja da je *$i* manje od 10. Pri svakom krugu petlje (izvršavanju naredbi iz tijela petlje), najprije se ispiše vrijednost varijable *$i* (zajedno s razmakom), a nakon toga se varijabla $i poveća za 1. Kad se varijabla $i deseti put poveća za 1, njena vrijednost će iznositi 10 i uvjet *$i* < 10 više neće vrijediti. Zbog toga, naredba za ispis vrijednosti više neće biti izvršena.

Moguće je pojavljivanje tzv. beskonačne petlje, slučaja u kojem izvršavanje petlje neće nikada prestati, jer uvjet ponavljanja neće nikada prestati vrijediti. Primjer jednostavne beskonačne petlje:

```
while (TRUE)
{
  echo "Petlja se izvršava";
}
```

Budući da je uvjet ponavaljanja uvijek istinit, ova petlja se nikada ne bi prestala izvršavati.

5.2. Petlja *do...while*

Ovaj tip petlje je sličan petlji *while*. Naredbe u tijelu petlje izvršavat će se ako je ispunjen uvjet ponavljanja, s tom razlikom da će se kod petlje *do...while* izvršiti barem jedanput, čak i ako uvjet nije ispunjen. Razlog tome je što se uvjet ponavljanja ne provjerava prije izvršavanja naredbi iz tijela petlje, već poslije njega. Oblik pisanja petlje *do...while* je ovakav:

```
do
{
  naredba1;
  naredba2;
} while (uvjet);
```

Primjer ispisivanja brojeva od 0 do 9 korištenjem petlje do...while izgledat će ovako:

```
<?php
  $i = 0;
  do
  {
    echo $i . " ";
    $i++;
  } while($i < 10);
?>
```

```
0 1 2 3 4 5 6 7 8 9
```

Ako obrnemo uvjet ponavljanja petlje tako da sada *$i* treba biti veći, a ne manji od 10, naredbe iz tijela petlje će se svejedno izvršiti jedanput:

```
<?php
  $i = 0;
  do
  {
    echo $i . " ";
    $i++;
  } while($i > 10);
?>
```

```
0
```

5.3. Petlja *for*

Petlja *for* ima poseban način pisanja.

```
for (pocetniIzraz; uvjet; ponavljajuciIzraz)
{
  naredba1;
  naredba2;
}
```

U zagradi nakon ključne riječi *for* nalaze se tri izraza odvojena znakom **;** .

- *pocetniIzraz* se izvršava prije prvog kruga petlje (krug petlje je jedno izvršavanje naredbi iz tijela petlje)
- *uvjet* je uvjet ponavljava i provjerava se prije svakog kruga petlje
- *ponavljajuciIzraz* se izvršava nakon svakog kruga petlje

Svaki od ta tri izraza može se izostaviti. Ako se izostavi *uvjet*, petlja će se izvršavati zauvijek. Izrazi *pocetniIzraz* i *ponavljajuciIzraz* mogu sadržavati više naredbi, a u tom slučaju one su odvojene zarezima.

Primjer ispisivanja brojeva od 0 do 9 pomoću petlje for izgledat će ovako:

```
<?php
  for ($i = 0; $i < 10; $i++)
  {
    echo $i . " ";
  }
?>
```

```
0 1 2 3 4 5 6 7 8 9
```

Petlja *for* najčešće se koristi kada je broj ponavljanja petlje zadan. U takvim slučajevima se kao brojač koristi varijabla (u gornjem primjeru *$i*). U početnom izrazu se postavlja početna vrijednost brojača (0), a u uvjetu konačna vrijednost brojača (10). U ponavljajućem izrazu se obavlja povećanje (ili ponekad smanjivanje) vrijednosti brojača.

Iznos za koji se povećava brojač petlje naziva se još i korak petlje. Ovako bi izgledala petlja koja ima korak 2, odnosno kod koje se brojač povećava za 2 nakon svakog kruga petlje:

```
<?php
  for ($i = 0; $i < 10; $i=$i+2)
  {
    echo $i . " ";
  }
?>
```

```
0 2 4 6 8
```

Ova petlja će ispisati samo parne brojeve između 0 i 9.

5.4. Petlja *foreach*

Petlje se često koriste za ispis članova polja. Kod polja s brojčanim ključem, članovi polja se mogu ispisati pomoću petlje *for*, *while* ili do...*while*. Primjer ispisa članova polja korištenjem petlje *for*:

```
<?php
  $gradovi = array("Zagreb","Split","Rijeka");

  for ($i = 0; $i < 3; $i++)
  {
    echo $gradovi[$i] . " ";
  }
?>
```

```
Zagreb Split Rijeka
```

Varijabla *$i*, koja se povećava za 1 u svakom krugu petlje, koristi se kao indeks za pristup članovima polja, tako da se u svakom krugu petlje ispiše po jedan član polja.

Za ispis članova polja može se koristiti i petlja *foreach*, čija je svrha upravo to. Oblik petlje *foreach* je ovakav:

```
foreach ($polje as $vrijednost)
{
  naredba1;
  naredba2;
}
```

Ova petlja će obaviti naredbe unutar tijela petlje jedanput za svakog člana polja *$polje*, a vrijednost trenutnog člana polja bit će u varijabli *$vrijednost*.

Moguće je dobiti i ključ trenutnog člana tako da se petlja *foreach* napiše na ovakav način:

```
foreach ($polje as $kljuc => $vrijednost)
{
  naredba1;
  naredba2;
}
```

Ključ trenutnog člana bit će u varijabli *$kljuc*.

Na ovaj način bi se ispisali članovi polja korištenjem petlje *foreach*:

```
<?php
  $gradovi = array("Zagreb","Split","Rijeka");

  foreach ($gradovi as $grad)
  {
    echo $grad . " ";
  }
?>
```

```
Zagreb Split Rijeka
```

Ako je uz vrijednost člana potrebno ispisati i njegov ključ, to je moguće učiniti kao u ovom primjeru:

```
<?php
  $post_br = array ("Zagreb" => 10000,
                    "Split" => 21000,
                    "Rijeka" => 51000);

  foreach ($post_br as $naziv => $broj)
  {
    echo "$broj $naziv <br />";
  }
?>
```

```
10000 Zagreb
21000 Split
51000 Rijeka
```

Upotrebom petlje *foreach* na ovaj način nije moguće izmijeniti vrijednost člana polja. Razlog tome je što se u svakom krugu petlje stvara kopija člana polja i smješta u varijablu *$broj*. Izmjena te varijable neće se odraziti na vrijednost člana polja.

U sljedećem primjeru povećanje svih članova polja za 1 neće uspjeti. Nakon što se izvrši petlja, vrijednosti članova polja bit će nepromijenjene:

```
<?php
  $brojevi = array(0,1,2,3,4);

  foreach ($brojevi as $broj)
  {
    echo $broj++;
  }
?>
```

Da bi se izmijenila vrijednost člana polja, potrebno je u petlji *foreach* rabiti referencu na njega, a ne kopiju vrijednosti. Referenca se dobiva pomoću operatora & i označava da varijabla pokazuje na originalnu vrijednost, te da će se zbog toga izmjena napravljena korištenjem reference odraziti na originalnu vrijednost.

U sljedećem primjeru koristi se referenca da bi se izmijenila originalna vrijednost člana polja. U ovom slučaju, izmjena će se odraziti na originalnom polju:

```
<?php
  $brojevi = array(0,1,2,3,4);

  foreach ($brojevi as &$broj)
  {
    echo $broj++;
  }
?>
```

Korištenje reference na člana polja u petlji *foreach* moguće je tek od PHP-ove verzije 5.

5.5. Ugnježđivanje petlji

Kao i uvjetne strukture, i petlje se mogu ugnježđivati jedne u drugu. Unutrašnja petlja može se promatrati kao zaseban blok kôda. Sljedeći primjer pokazujeda vanjska petlja 3 puta poziva izvršavanje unutrašnje petlje, koja se izvršava 5 puta. To znači da se naredba za ispis koja se nalazi u tijelu unutrašnje petlje, izvršava 3 x 5 = 15 puta.

```
<?php
  for ($i = 1; $i <= 3; $i++)
  {
    for ($j = 1; $j <= 5; $j++)
    {
      echo "$i.$j ";
    }
    echo "<br />";
  }
?>
```

```
1.1 1.2 1.3 1.4 1.5
2.1 2.2 2.3 2.4 2.5
3.1 3.2 3.3 3.4 3.5
```

Naredba za ispis svaki put ispisuje vrijednost brojača vanjske petlje i vrijednost brojača unutrašnje petlje odvojene točkom.

Pomoću dvije ugnježđene petlje mogu se ispisati članovi dvodimenzionalnog polja:

```
<?php
  $igra = array( array ("O", "O", ""),
                 array ("O", "X", "O"),
                 array ("X", "O", "X") );

  for ($i = 0; $i < 3; $i++)
  {
    for ($j = 0; $j < 3; $j++)
    {
      echo igra[$i][$j] . " ";
    }
    echo "<br />";
  }
?>
```

```
O O
O X O
X O X
```

Dvodimenzionalno polje može se ispisati i pomoću dvije ugnježđene *foreach* petlje:

```
<?php
  $igra = array( array ("O", "O", ""),
                 array ("O", "X", "O"),
                 array ("X", "O", "X") );

  foreach ($igra as $redak)
  {
    foreach ($redak as $clan)
    {
      echo $clan . " ";
    }
    echo "<br />";
  }
?>
```

```
O O
O X O
X O X
```

5.6. Prijevremeni izlazak iz petlje

Ponekad je, ako je neki uvjet ispunjen, potrebno prijevremeno završiti s izvođenjem petlje. Izvršavanje petlje može se prekinuti navođenjem ključne riječi *break*.

U ovom primjeru, petlja bi trebala ispisati brijeve od 0 do 9, ali se, zbog ključne riječi *break*, izvršavanje petlje prekida nakon što se *$i* poveća na 6:

```
<?php
  for ($i = 0; $i < 10; $i++)
  {
    if ($i == 6)
    {
      break;
    }
    echo $i . " ";
  }
?>
```

```
0 1 2 3 4 5
```

Također je moguće i preskakanje ostatka naredbi u petlji, i nastavak izvođenja petlje od sljedećeg kruga. Tome služi ključna riječ *continue*:

```
<?php
  for ($i = 0; $i < 10; $i++)
  {
    if ($i == 6)
    {
      continue;
    }
    echo $i . " ";
  }
?>
```

```
0 1 2 3 4 5 7 8 9
```

U gornjem primjeru će tako biti preskočen ispis broja 6, ali će se svi brojevi nakon njega ispisati.

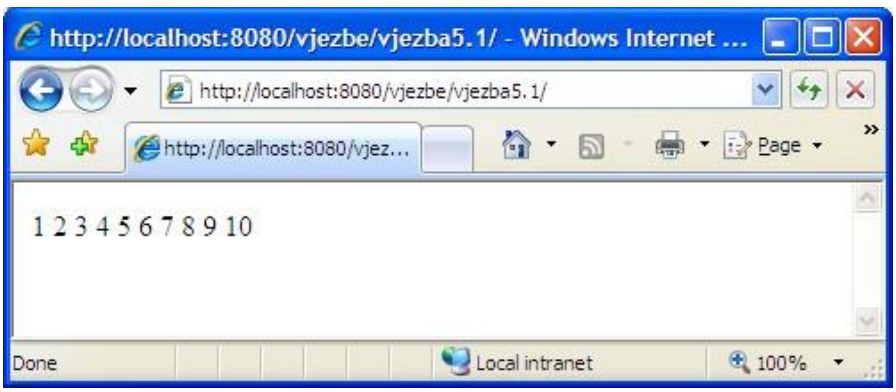
Vježba 5.1 – Korištenje petlje *for*

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *index.php*.
2. U datoteku upišite oznake za PHP kôd i unutar njih petlju koja će ispisati brojeve od 1 do 10:

```
<?php
  for ($i = 1; $i<=10; $i++)
  {
    echo $i . " ";
  }
?>
```

   Spremite datoteku u mapu "..\D350\vjezbe\vjezba5.1" .

3. Pokrenite servis *EasyPHP*, ako već nije pokrenut.
4. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba5.1/*



5. Unutar petlje, umjesto naredbe *echo*, dodajte još jednu petlju čiji brojač također ide od 1 do 10. U tijelu unutrašnje petlje napišite naredbu koja će ispisati umnožak brojača vanjske petlje ($i) i brojača unutrašnje petlje ($j):

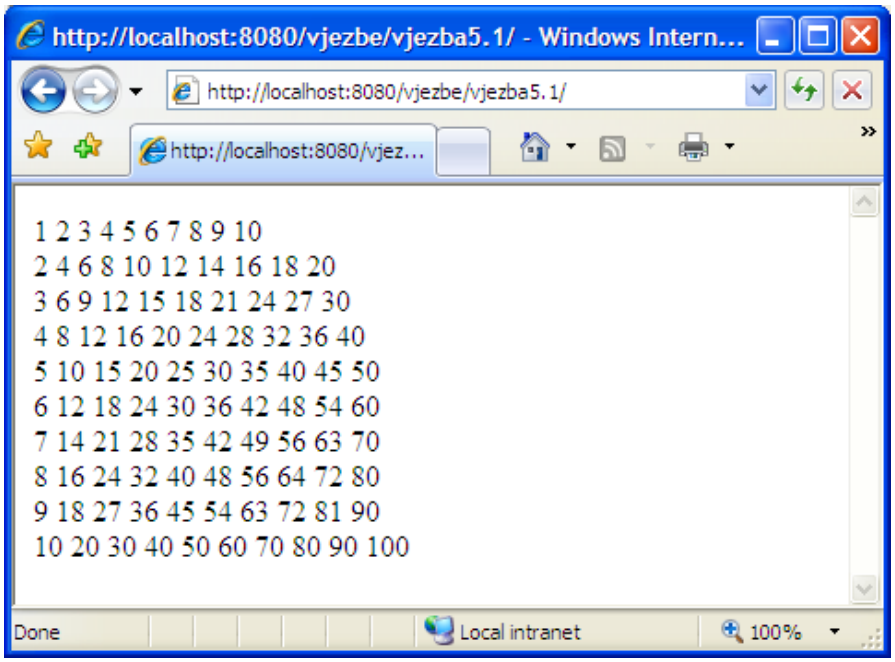
```
<?php
  for ($i = 1; $i <= 10; $i++)
  {
    for ($j = 1; $j <= 10; $j++)
    {
      echo $i*$j . " ";
    }
  }
?>
```

6. U tijelo vanjske petlje, dodajte naredbu echo koja će ispisati HTML oznaku za novi red:

```
<?php
  for ($i = 1; $i <= 10; $i++)
  {
    for ($j = 1; $j <= 10; $j++)
    {
      echo $i*$j . " ";
    }
    echo "<br />";
  }
?>
```

Spremite datoteku.

7. Osvježite stranicu u web pregledniku.



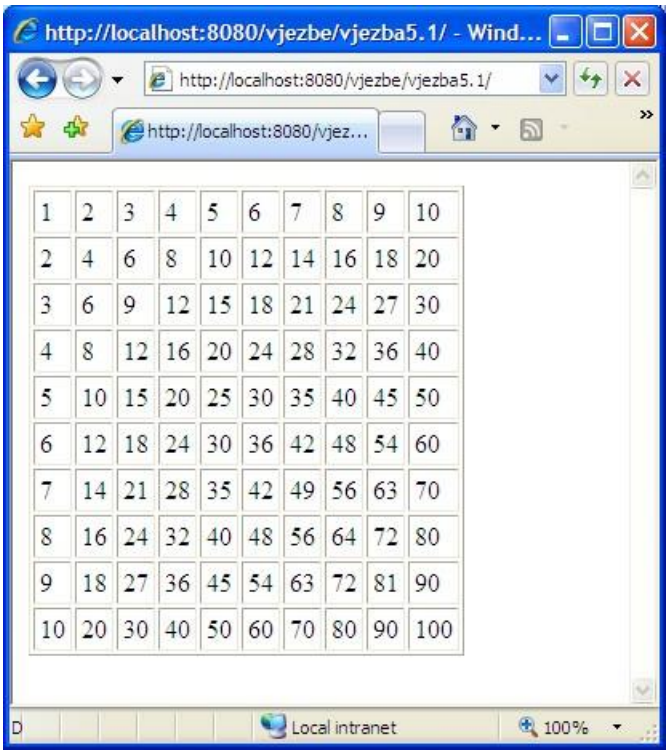
Dobiveni rezultat je zapravo tablica množenja za brojeve od 1 do 10.

8. Izmijenite kôd tako da tablicu množenja ispišete pomoću HTML tablice:

```
<table border="1" cellpadding="3">
<?php
  for ($i = 1; $i <= 10; $i++)
  {
    echo "<tr>";
    for ($j = 1; $j <= 10; $j++)
    {
      echo "<td>";
      echo $i*$j;
      echo "</td>";
    }
    echo "</tr>";
  }
?>
</table>
```

Spremite datoteku.

9. Osvježite stranicu u web pregledniku.



1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Vježba 5.2 – Ispis članova polja

Ova vježba nastavak je vježbe 4.2. iz prethodnog poglavlja.

U ovoj se vježbi kôd iz vježbe 4.2 mijenja tako da se za ispisivanje članova polja i računanje prosjeka ocjena koristi petlja *foreach*.

1. U programu *PHP Designer* otvorite datoteku *index.php* u mapi "..\D350\vjezbe\vjezba5.2" .

2. Izbrišite ove naredbe (dakle, sve naredbe osim naredbe koja stvara polje *$ocjene*):

```
  $prosjek = ($ocjene["Hrvatski"] + $ocjene["Matematika"]
              $ocjene["Povijest"]) / 3;

  echo "Hrvatski: {$ocjene["Hrvatski"]}<br />";
  echo "Matematika: {$ocjene["Matematika"]}<br />";
  echo "Povijest: {$ocjene["Povijest"]}<br />";
  echo "Prosjek ocjena je: $prosjek.";
?>
```

3. Umjesto njih, za računanje prosjeka i ispis ocjena (i naziva predmeta), koristi se petlja *foreach*:

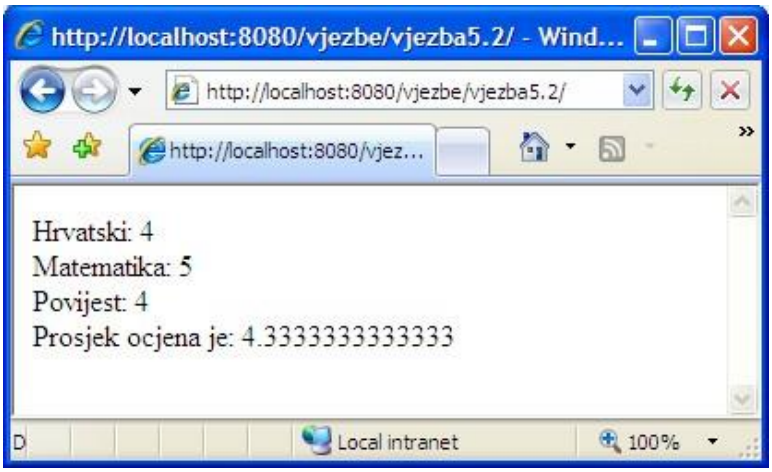
```
  $zbroj = 0;
  foreach ($ocjene as $predmet => $ocjena)
  {
    echo "$predmet: $ocjena <br/>";
    $zbroj += $ocjena;
  }

  echo "Prosjek ocjena je: " . $zbroj/3;
?>
```

   Prosjek se računa tako da se u svakom krugu petlje varijabli *$zbroj* (koja je prvotno postavljena na 0) dodaje iznos ocjene. Nakon što petlja završi, ukupni zbroj se dijeli s 3 i dobiva se prosjek.

4. Spremite datoteku.

5. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

6. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba5.2/*



Konačni ispis ocjena i prosjeka izgleda isto kao i u vježbi 4.2, no ostvaren je jednostavnije i lakše, pomoću petlje *foreach*.

Ova vježba nastavak je vježbe 4.3. iz prethodnog poglavlja.

Vježba 5.3 – Ispis članova dvodimenzionalnog polja

U ovoj se vježbi kôd iz vježbe 4.3 mijenja tako da se umjesto pisanja sličnih naredbi i odlomaka kôda tri puta, za ispisivanje članova dvodimenzionalnog polja i računanje prosjeka ocjena koristi petlja *foreach*.

1. U programu *PHP Designer* otvorite datoteku *index.php* u mapi "..\D350\vjezbe\vjezba5.3" .

2. Izbrišite naredbe koje ispisuju pojedine ocjene (masno otisnute naredbe):

```
  $prosjekIvica = ( $ocjene["Ivica"]["Hrvatski"] +
                    $ocjene["Ivica"]["Matematika"] +
                    $ocjene["Ivica"]["Povijest"] ) / 3;

  $prosjekTomica = ( $ocjene["Tomica"]["Hrvatski"] +
                     $ocjene["Tomica"]["Matematika"] +
                     $ocjene["Tomica"]["Povijest"] ) / 3;

  $prosjekPerica = ( $ocjene["Perica"]["Hrvatski"] +
                     $ocjene["Perica"]["Matematika"] +
                     $ocjene["Perica"]["Povijest"] ) / 3;
?>
```

3. Zaglavlje HTML tablice ispisivat ćemo na isti način kao i prije, ali za ispis redaka koristit ćemo se petljom *foreach*. Izbrišite HTML kôd (zajedno s umetnutim PHP kôdom) koji služi za ispis redaka tablice:

```
    <tr>
      <th>Ivica</th>
      <td><?php echo $ocjene["Ivica"]["Hrvatski"] ?></td>
      <td><?php echo $ocjene["Ivica"]["Matematika"] ?></td>
      <td><?php echo $ocjene["Ivica"]["Povijest"] ?></td>
      <td><?php echo $prosjekIvica ?></td>
    </tr>
    <tr>
      <th>Tomica</th>
      <td><?php echo $ocjene["Tomica"]["Hrvatski"] ?></td>
      <td><?php echo $ocjene["Tomica"]["Matematika"] ?></td>
      <td><?php echo $ocjene["Tomica"]["Povijest"] ?></td>
      <td><?php echo $prosjekTomica ?></td>
    </tr>
    <tr>
      <th>Perica</th>
      <td><?php echo $ocjene["Perica"]["Hrvatski"] ?></td>
      <td><?php echo $ocjene["Perica"]["Matematika"] ?></td>
      <td><?php echo $ocjene["Perica"]["Povijest"] ?></td>
      <td><?php echo $prosjekPerica ?></td>
    </tr>
```

4. Na mjesto HTML kôda izbrisanog u prethodnom koraku napišite, unutar oznaka za PHP kôd, dvije ugnježđene petlje *foreach*:

```
      <th>Prosjek</th>
    </tr>
<?php
    foreach ($ocjene as $ime => $ucenik)
    {
      echo "<tr>";
      echo "<th>$ime</th>";
      $zbroj = 0;
      foreach ($ucenik as $ocjena)
      {
        echo "<td>$ocjena</td>";
        $zbroj += $ocjena;
      }
      echo "<td>" . $zbroj/3 . "</td>";
      echo "</tr>";
    }
?>
</table>
```

Vanjska petlja *foreach* izvršava se za svaki redak dvodimenzionalnog polja *$ocjene* (dakle za svakog učenika). Ključ retka (ime učenika) sprema se u varijablu *$ime*, dok se sam redak sprema u varijablu *$ucenik.*

Varijabla *$ucenik* je zapravo jednodimenzionalno polje koje sadrži ocjene jednog učenika. Pomoću unutrašnje petlje *foreach* kreće se po njegovim članovima, koji se spremaju u varijablu *$ocjena*.

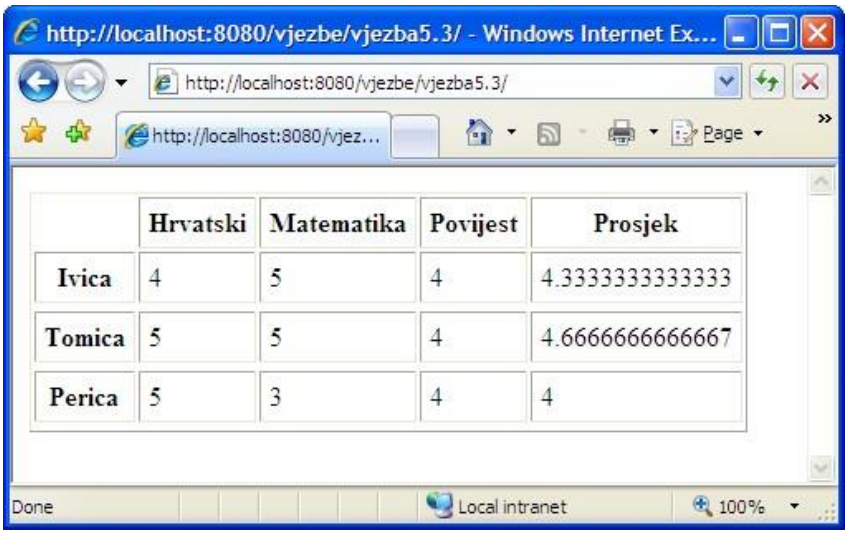
Vanjska petlja ispisuje HTML oznaku za redak (*tr*), zatim ime učenika (unutar oznaka *th*), a unutrašnja petlja ispisuje učenikove ocjene (unutar oznaka *td*). U unutrašnjoj petlji računa se ukupan zbroj

učenikovih ocjena koji se zatim (ponovno u vanjskoj petlji) dijeli s brojem ocjena (da se dobije prosjek) i zatim ispisuje (unutar oznaka *td*). U vanjskoj petlji se na kraju zatvara redak pomoću završne oznake *tr*.

5. Spremite datoteku.

6. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

7. U web pregledniku upišite adresu: *http://localhost/vjezbe/vjezba5.3/*



	Hrvatski	Matematika	Povijest	Prosjek
Ivica	4	5	4	4.3333333333333
Tomica	5	5	4	4.6666666666667
Perica	5	3	4	4

Dobiven je isti rezultat kao i u vježbi 4.3, ali je korištenjem petlje *foreach* izbjegnuto višestruko pisanje sličnih naredbi.

U ovom je poglavlju obrađeno:

- koncept petlje
- petlja *while*
- petlja *do...while*
- petlja *for*
- petlja *foreach* i njeno korištenje za ispis članova polja
- ugnježđivanje petlji
- izlazak iz petlje pomoću naredbe *break* i preskakanje trenutnog kruga petlje pomoću naredbe *continue*

6. Funkcije

Naredbe koje se koriste više puta mogu se napisati unutar funkcije. Svaki put kad je potrebno izvršiti te naredbe, poziva se funkcija. Izdvajanjem dijelova kôda u funkcije taj je kôd potrebno napisati samo jedanput, nema potrebe pisati ga više puta. Time se postiže ušteda vremena, manja je mogućnost pogreške i jednostavnije je kasnije mijenjanje kôda.

Funkcije mogu primati određene podatke (argumente funkcije), kojima će se koristiti u svom radu. Funkcije također mogu vratiti izračunatu vrijednost kao rezultat.

6.1. Funkcije

Funkcije se pišu tako da se navede ključna riječ *function*, nakon koje dolazi ime funkcije i par okruglih zagrada. Poslije njih, u vitičastim zagradama pišu se naredbe koje će funkcija obaviti:

```
function imeFunkcije()
{
  naredba1;
  naredba2;
}
```

Za imenovanje funkcija vrijede ista pravila kao i za imenovanje varijabli: mogu se koristiti slova, brojevi i povlaka (_), a ime funkcije može početi slovom ili povlakom. Ime funkcije ne smije biti jednako imenu već postojeće funkcije u istoj datoteci, imenu ugrađene funkcije ili ključnoj riječi PHP-a.

Imena funkcija, za razliku od imena varijabli, nisu osjetljiva na mala i velika slova, tako da će *MojaFunkcija* i *mojafunkcija* biti prepoznate kao ista funkcija. Preporuča se odabrati jedan način imenovanja funkcija i pridržavati ga se.

Jednostavna funkcija koja će ispisati neki tekst izgledat će ovako:

```
<?php
  function IspisiPozdrav()
  {
    echo "Pozdrav, svijete!";
  }
?>
```

Poziv te funkcije izgledat će ovako:

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

```
"Pozdrav, svijete!"
```

Da bi se funkcija mogla pozvati, definicija funkcije mora se nalaziti u istoj datoteci kao i poziv funkcije. Nije nužno da je definicija funkcije u datoteci napisana prije njenog poziva.

6.2. Funkcije s argumentima

Prilikom poziva, funkciji se mogu predati određeni podaci koje će ona upotrijebiti. Ti podaci se nazivaju **argumentima** ili parametrima funkcije i moraju se navesti prilikom definicije funkcije:

```
function imeFunkcije($argument1, $argument2)
{
  naredba1;
  naredba2;
}
```

Prilikom navođenja argumenata navodi se samo njihovo ime, a ne navodi se tip podatka.

Primjer funkcije koja prima dva argumenta:

```
<?php
  function Mnozi($a, $b)
  {
    echo $a * $b;
  }
?>
```

Ova funkcija pomnožit će dva zadana broja i ispisati njihov umnožak. Njen poziv izgledat će ovako:

```
<?php
  Mnozi(4,2);
?>
```

```
8
```

Predefinirani argumenti

Pri pozivu funkcije moraju se uvijek navesti svi argumenti. Iznimka od toga su predefinirani argumenti čije se vrijednosti zadaju prilikom definicije funkcije.

U ovom primjeru se kao predefinirana vrijednost argumenta *$b* zadaje 2:

Prilikom definicije funkcije, predefinirani argumenti se navode tek nakon što se navedu svi "obični" argumenti.

```
<?php
  function Mnozi($a, $b = 2)
  {
    echo $a * $b;
  }
?>
```

Predefinirani argumenti funkcije mogu se, ali i ne moraju navesti prilikom poziva.

Ako se predefinirani argument ne navede, koristi se njegova unaprijed zadana vrijednost:

```
<?php
  Mnozi(4);
?>
```

```
8
```

U gornjem primjeru se, ako se ne navede drugi argument, predani broj množi s 2.

Prijenos argumenata po vrijednosti

Najčešći način prijenosa argumenata funkciji je prijenos po vrijednosti kod kojeg se vrijednost argumenta kopira u novu varijablu.

U sljedećem primjeru, funkcija *Povecaj* povećava predanu vrijednost za 1. Moglo bi se očekivati da će nakon povratka iz funkcije vrijednost varijable *$broj* biti uvećana za 1, ali to se ne događa.

```
<?php
  function Povecaj($a)
  {
    $a++;
    echo "Vrijednost u funkciji: $a <br />";
  }

  $broj = 2;
  Povecaj($broj);
  echo "Vrijednost nakon povratka iz funkcije: $broj";
?>
```

```
Vrijednost u funkciji: 3
Vrijednost nakon povratka iz funkcije: 2
```

Razlog tome je što funkcija dobiva kopiju vrijednosti, a ne originalnu varijablu, te izmjena varijable u funkciji nema utjecaja na vrijednost originalne varijable.

Prijenos argumenata po referenci

Stvaranjem **reference** na varijablu, za što se koristi operator **&**, postiže se da obje varijable (referencirana i originalna varijabla) sadrže istu vrijednost Izmjena jedne varijable rezultirat će promjenom vrijednosti obje varijable.

```
<?php
  $a = 1;
  $b = &$a;
  echo "$a $b <br/>";

  $a++;
  echo "$a $b <br/>";

  $b++;
  echo "$a $b";
?>
```

```
1 1
2 2
3 3
```

U drugoj naredbi stvara se varijabla *$b* kao referenca na varijablu *$a*. Nakon toga, obje varijable pokazuju na istu vrijednost, i promjena na bilo kojoj od njih utječe na drugu varijablu.

Reference se mogu koristiti prilikom prijenosa argumenata u funkciju. Umjesto da se funkciji preda kopija argumenta, funkciji se predaje referenca koja pokazuje na originalnu vrijednost. U tom slučaju, izmjene napravljene na argumentu vrijedit će i nakon povratka iz funkcije.

Ako se unutar funkcije želi izmijeniti originalna vrijednost argumenta, potrebno je umjesto vrijednosti argumenta predati referencu na njega.

Operator referenciranja (**&**) potrebno je navesti samo prilikom definicije funkcije, a ne i prilikom njenog poziva.

Funkcija *Povecaj* u ovom će primjeru izmijeniti i originalnu varijablu *$broj*:

```
<?php
  function Povecaj(&$a)
  {
    $a++;
    echo "Vrijednost u funkciji: $a <br />";
  }

  $broj = 2;
  Povecaj($broj);
  echo "Vrijednost nakon povratka iz funkcije: $broj";
?>
```

```
Vrijednost u funkciji: 3
Vrijednost nakon povratka iz funkcije: 3
```

6.3. Vraćanje rezultata funkcije

Uz vraćanje rezultata na način da se promijeni argument prenesen po referenci, postoji i češće korišteni način vraćanja rezultata funkcije, upotrebom ključne riječi *return*.

Funkcija koja množi dva broja i vraća rezultat množenja pomoću ključne riječi *return*, izgledat će ovako:

```
<?php
  function Mnozi($a, $b)
  {
    return $a * $b;
  }
?>
```

Poziv te funkcije izgledat će ovako:

```
<?php
  $c = Mnozi(3,5);
  echo $c;
?>
```

```
15
```

Ključna riječ *return* označava izlazak iz funkcije. Ako se iza nje nalazi još neka naredba, ona neće biti izvršena. Ključna riječ *return* može se koristiti i za izlazak iz funkcije bez vraćanja rezultata:

```
<?php
  function IspisiTekst($tekst)
  {
    if ($tekst == "")
    {
      return;
    }
    echo $tekst;
  }
?>
```

Ako se funkciji *IspisiTekst* u gornjem primjeru preda prazan znakovni niz, ona neće ispisati ništa jer će se izvršavanje funkcije prekinuti prije naredbe *echo*.

6.4. Ugnježđivanje funkcija

Funkciju je moguće pozvati iz druge funkcije. U sljedećem primjeru, funkcija *Kvadriraj* ne obavlja množenje sama, već se za to koristi funkcijom *Mnozi*.

```
<?php

  function Mnozi($a, $b)
  {
    return $a * $b;
  }

  function Kvadriraj($a)
  {
    return Mnozi($a, $a);
  }

  $c = Kvadriraj(5);
?>
```

```
25
```

Funkcija može pozvati samu sebe. Takva funkcija naziva se **rekurzivnom funkcijom**. Klasičan primjer takve funkcije je računanje matematičke funkcije faktorijela koja se računa po formuli $n! = n * (n - 1)!$.

Kod rekurzivne funkcije je bitno postojanje tzv. početnog slučaja. Ako je on ispunjen, funkcija ne poziva opet samu sebe, već vraća rezultat. U primjeru s faktorijelom to je slučaj kad je *n* jednako 0 ili 1.

```
<?php
  function Faktorijela($n)
  {
    if ($n == 0 || $n == 1)
    {
      return 1;
    }
    else
    {
      return n * Faktorijela(n-1);
    }
  }
?>
```

6.5. Uključivanje vanjskih datoteka u kôd

Potreba za korištenjem neke funkcije može se pojaviti na više mjesta, i to u odvojenim skriptama. Da se funkcije ne bi morale ponovno pisati za svaku skriptu koja ih rabi, moguće ih je napisati u zasebnoj datoteci, i onda se ta datoteka uključuje u kôd skripte koja treba pozvati funkciju.

Funkcija za množenje dva broja može se napisati u zasebnoj datoteci koja se može zvati *mnozenje.php*:

```
<?php
  function Mnozi($a, $b)
  {
    return $a * $b;
  }
?>
```

Da bi se funkcija pozvala u nekoj drugoj skripti, potrebno je upotrebom naredbe *include* u kôd trenutne skripte uključiti datoteku u kojoj se funkcija nalazi:

```
<?php
  include("mnozenje.php");

  $c = Mnozi(3, 4);
?>
```

Datoteka se može uključiti i upotrebom naredbe *require*:

```
<?php
  require("mnozenje.php");

  $c = Mnozi(3, 4);
?>
```

Razlika između naredbi *include* i *require* je u načinu reagiranja na pogreške. Ako se dogodi pogreška prilikom izvršavanja naredbi iz datoteke uključene pomoću naredbe *include*, ispisat će se poruka o pogrešci, ali će skripta nastaviti s izvođenjem. Kod naredbe *require* u takvom slučaju bi se prekinulo i izvršavanje skripte u koju je datoteka uključena.

Preporuča se korištenje naredbe *require* za uključivanje vanjskih datoteka u kôd, zbog toga što u slučaju pogreške prestaje izvršavanje skripte.

Upotreba naredbi *include* i *require* nije ograničena samo na uključivanje datoteka koje sadrže funkcije. Pomoću njih moguće je uključiti datoteke koje sadrže PHP naredbe, HTML kôd ili običan tekst. Bitno je zapamtiti da će sadržaj uključene datoteke biti umetnut u skriptu na mjesto poziva naredbe *include*, odnosno *require*.

6.6. Doseg varijabli

Doseg ili vidljivost varijable određuje na kojim će mjestima unutar skripte biti dostupna varijabla, odnosno vrijednost koju sadržava. Varijable po njihovom dosegu moguće je podijeliti na tri vrste:

- lokalne varijable
- globalne varijable
- predefinirane globalne varijable (superglobalne varijable).

Lokalne varijable

Lokalne varijable vidljive su samo unutar funkcije u kojoj se koriste. U sljedećem primjeru varijabla *$a* u funkciji neće imati veze s varijablom *$a* koja se koristi u ostatku skripte.

```
<?php
  function Ispisi()
  {
    $a = 2;
    echo $a;
  }

  $a = 1;
  Ispisi();
?>
```

```
2
```

Pod lokalne varijable spadaju i argumenti funkcije, koji neće imati veze s eventualnim istoimenim varijablama u ostatku skripte.

Globalne varijable

Globalne varijable su varijable definirane izvan funkcija. One su vidljive unutar cijele datoteke (i svih uključenih datoteka), ali ne i unutar funkcija.

Ako se globalna varijabla želi koristiti unutar funkcije, potrebno ju je definirati kao globalnu varijablu pomoću ključne riječi *global*.

U PHP-u nema deklariranja varijable (navođenja tipa podataka koji varijabla koristi). Pod definiranjem varijable ovdje se smatra prvo pridjeljivanje vrijednosti varijabli.

```
<?php
  function Ispisi()
  {
    global $a;
    echo $a;
  }

  $a = 1;
  Ispisi();
?>
```

```
1
```

Predefinirane globalne varijable

Predefinirane globalne varijable, koje se nazivaju i superglobalnim varijablama, su polja dostupna na bilo kojem mjestu u svim skriptama.

U polju *$GLOBALS*, koje je dostupno u svim skriptama, nalazit će se globalne varijable definirane u trenutnoj skripti. Pojedinom članu tog polja pristupa se preko znakovnog ključa – imena varijable.

Pomoću tog polja može se pristupiti globalnim varijablama i unutar funkcije:

```
<?php
  function Ispisi()
  {
    echo $GLOBALS["a"];
  }

  $a = 1;
  Ispisi();
?>
```

```
1
```

Osim polja *$GLOBALS* postoje i druge predefinirane globalne varijable. Radi se poljima koja sadrže vrijednosti vezane za postavke poslužitelja, HTTP zahtjeva i slično. Ova polja također imaju znakovne ključeve preko kojih se pristupa pojedinoj vrijednosti.

U globalno dostupnom polju *$_SERVER* nalaze se vrijednosti vezane za web poslužitelja. Pomoću njega, u svakoj se skripti može pristupiti IP adresi poslužitelja i portu kojim se poslužitelj koristi:

```
<?php
  echo $_SERVER["SERVER_ADDR"];
  echo "<br/>";
  echo $_SERVER["SERVER_PORT"];
?>
```

```
127.0.0.1
80
```

Ako se ovaj primjer pokrene na lokalnom računalu, bit će ispisana IP adresa 127.0.0.1, što je adresa lokalnog web poslužitelja. Ako je on instaliran na portu 80, bit će ispisana vrijednost 80.

U sljedećoj tablici dan je pregled predefiniranih globalno dostupnih polja:

Polje	Opis
$GLOBALS	vrijednosti globalnih varijabli iz trenutne skripte
$_SERVER	vrijednosti postavljene od strane web poslužitelja
$_ENV	vrijednosti iz okruženja u kojem je instaliran PHP
$_GET	vrijednosti dostupne u URL-u trenutne stranice
$_POST	vrijednosti prenešene u HTTP zahtjevu metodom *POST*
$_COOKIE	vrijednosti zapisane u kolačićima (tekstualnim datotekama koje web preglednik sprema na korisnikovom računalu)
$_REQUEST	vrijednosti iz polja *$_GET, $_POST* i *$_COOKIE*
$_FILES	vrijednosti koje se odnose na datoteke poslane na poslužitelj

Pojedine dostupne superglobalne varijable ovise o instaliranom *web* poslužitelju, kao i o sadržaju trenutnog HTTP zahtjeva.

Vježba 6.1 – Funkcija

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *index.php*.
2. U datoteci napišite funkciju koja računa prosjek tri broja i vraća izračunati rezultat:

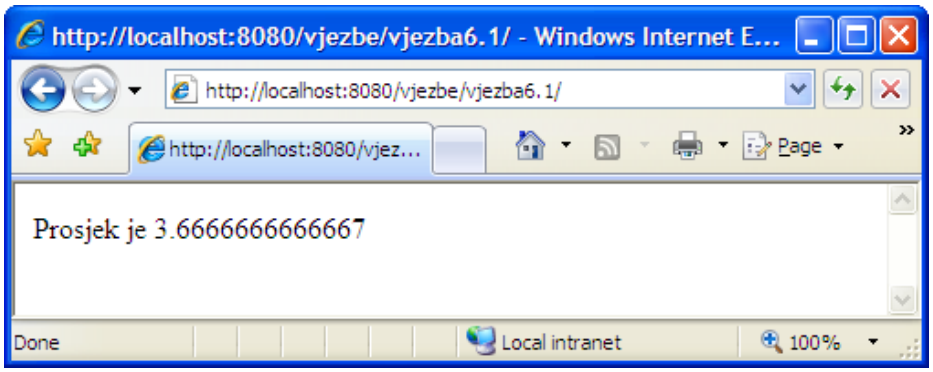
```
<?php
  function Prosjek($a, $b, $c)
  {
    return ($a + $b + $c) / 3;
  }
?>
```

3. Stvorite tri varijable, pridružite im vrijednosti i izračunajte njihov prosjek:

```
  }

  $a = 3;
  $b = 4;
  $c = 4;

  $d = Prosjek($a, $b, $c);
  echo "Prosjek je " . $d;
?>
```

Spremite datoteku u mapu "..\D350\vjezbe\vjezba6.1" .

4. Pokrenite servis *EasyPHP*, ako već nije pokrenut.
5. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba6.1/*



Ova vježba se nastavlja na vježbu 6.1. i vježbu 4.2 iz 4. poglavlja.

Vježba 6.2 – Korištenje funkcije

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *prosjek.php*.
2. U datoteku napišite funkciju *Prosjek* iz prethodne vježbe:

```
<?php
  function Prosjek($a, $b, $c)
  {
    return ($a + $b + $c) / 3;
  }
?>
```

   Spremite datoteku u mapu "..\D350\vjezbe\vjezba6.2" .

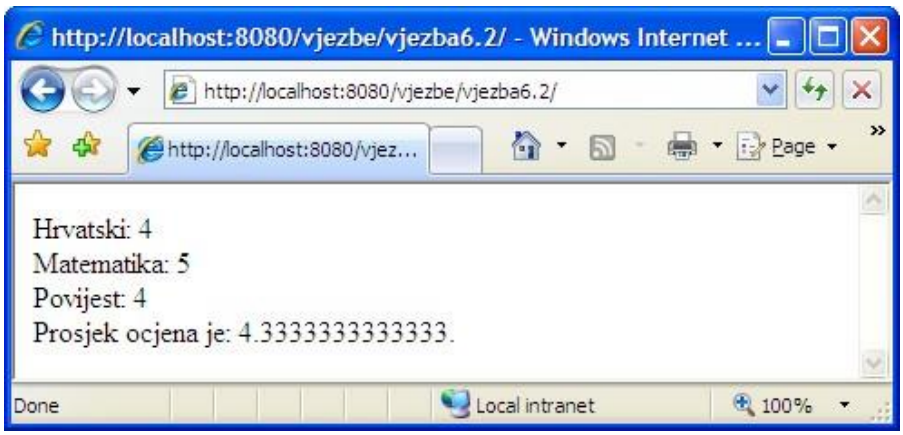
3. U programu *PHP Designer* otvorite datoteku *index.php* u mapi "..\D350\vjezbe\vjezba6.2".
4. Obrišite naredbu koja računa prosjek ocjena:

```
  $prosjek = ($ocjene["Hrvatski"] + $ocjene["Matematika"] +
              $ocjene["Povijest"]) / 3;
```

5. Na njeno mjesto dodajte naredbu koja će uključiti datoteku *prosjek.php* u kôd, i zatim naredbu koja će računati prosjek ocjena pomoću funkcije *Prosjek*:

```
  require("prosjek.php");

  $prosjek = Prosjek($ocjene["Hrvatski"], $ocjene["Matematika"],
                     $ocjene["Povijest"]);
```

6. Spremite datoteku.
7. Pokrenite servis *EasyPHP*, ako već nije pokrenut.
8. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba6.2/*



Dobiveni rezultat je isti kao i u vježbi 4.2, ali je ovdje prosjek ocjena izračunat pomoću funkcije koja računa prosjek tri broja.

Vježba 6.3 – Korištenje funkcije (2)

Ova vježba se nastavlja na vježbu 5.3. iz prethodnog poglavlja.

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *prosjek.php*.
2. U datoteku napišite funkciju *Prosjek* koja će računati prosjek vrijednosti članova predanog polja:

```
<?php
  function Prosjek ($polje)
  {
    $zbroj = 0;
    $i = 0;
    foreach ($polje as $broj)
    {
      $zbroj += $broj;
      $i++;
    }
    $prosjek = $zbroj / $i;
    return $prosjek;
  }
?>
```

Ova funkcija je bolja od one iz prethodne vježbe zato što nije ograničena na tri broja - može izračunati prosjek proizvoljnog broja članova polja.

Na početku se postavljaju vrijednosti varijabli *$zbroj* i *$i* na 0. Članovima polja prolazi se pomoću petlje *foreach*. Varijabla *$zbroj* koristi se za zbrajanje svih članova polja, a varijabla *$i* za brojanje koliko polje ima članova. Rezultat se dobiva dijeljenjem varijabli *$zbro*j i *$i*.

3. Spremite datoteku u mapu "..\D350\vjezbe\vjezba6.3\" .
4. U programu *PHP Designer* otvorite datoteku *index.php* u mapi "..\D350\vjezbe\vjezba6.3".
5. U datoteku dodajte naredbu koja će uključiti datoteku *prosjek.php* u kôd ove datoteke:

```
<?php
   require("prosjek.php");

   foreach ($ocjene as $ime => $ucenik)
```

6. Obrišite naredbe koje služe za računanje prosjeka ocjena u petlji jer više nisu potrebne (masno označene naredbe):

```
      $zbroj = 0;
      foreach ($ucenik as $ocjena)
      {
        echo "<td>$ocjena</td>";
        $zbroj += $ocjena;
      }
```

7. Naredbu koja ispisuje prosjek ocjena promijenite tako da sada poziva funkciju *Prosjek*:

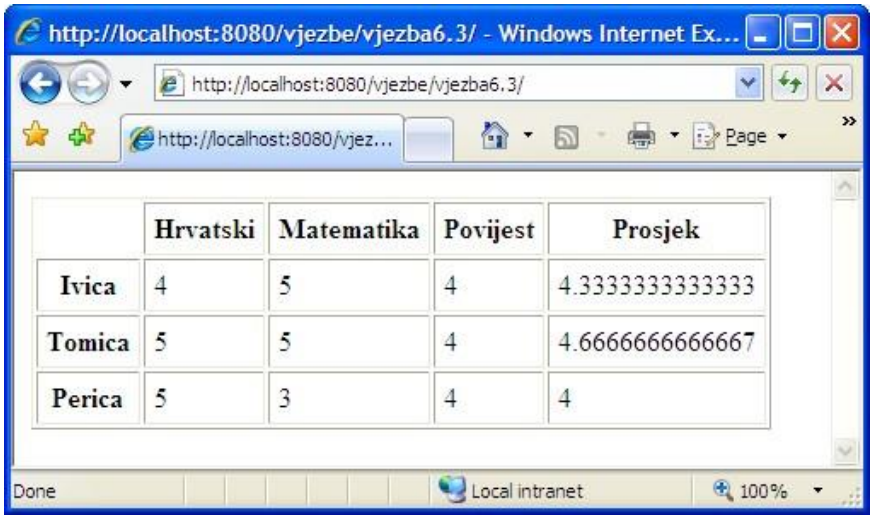
```
      foreach ($ucenik as $ocjena)
      {
        echo "<td>$ocjena</td>";
      }
      echo "<td>" . Prosjek($ucenik) . "</td>";
      echo "</tr>";
    }
}
```

   Funkciji *Prosjek* kao argument se predaje polje učenik, koje sadrži ocjene jednog učenika i predstavlja jedan redak dvodimenzionalnog polja koje sadrži ocjene svih učenika.

9. Spremite datoteku.

10. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

11. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba6.3/*



Dobiveni rezultat isti je kao i u vježbi 5.3. Prosjek ocjena izračunat je ovdje pomoću funkcije koja računa prosjek članova polja, a koja se poziva za svakog pojedinog učenika.

U ovom je poglavlju obrađeno:

- funkcije
- pozivanje funkcije
- funkcije s argumentima i prijenos vrijednosti u funkciju
- funkcije koje vraćaju rezultat
- uključivanje vanjskih datoteka u kôd
- doseg varijabli

7. Neke ugrađene funkcije PHP-a

Osim funkcija koje sam napiše, programer u svom kôdu može rabiti i već postojeće (ugrađene) funkcije jezika PHP. U PHP-u postoji velik broj korisnih funkcija, a u ovom poglavlju je dan kratak pregled samo nekih od njih.

7.1. Funkcije za rad sa znakovnim nizovima

Funkcija ***trim*** koristi se za micanje praznina s početka i kraja znakovnog niza. Pod prazninama se smatraju razmaci, tabulatori i znakovi za novi red. Funkcija vraća niz iz kojeg su maknute praznine ili zadani znakovi. Primjer korištenja funkcije *trim*:

```
<?php
  $niz = "\t O miševima i ljudima  \n";
  $niz = trim($niz);
  echo $niz;
?>
```

```
O miševima i ljudima
```

Funkcije ***strtoupper*** i ***strtolower*** služe za pretvaranje svih znakova u nizu u znakove napisane samo velikim, odnosno samo malim slovima. Primjer korištenja ovih funkcija:

```
<?php
  $niz = "O miševima i ljudima";

  $malaSlova = strtolower($niz);
  echo $malaSlova;
  echo "<br />";

  $velikaSlova = strtoupper($niz);
  echo $velikaSlova;
?>
```

```
o miševima i ljudima
O MIŠEVIMA I LJUDIMA
```

Funkcija ***strlen*** vraća duljinu zadanog niza. Primjer korištenja ove funkcije:

```
<?php
  $niz = "O miševima i ljudima";
  $duljina = strlen($niz);
  echo $duljina;
?>
```

```
20
```

Funkcija ***substr*** služi za dobivanje dijela ulaznog niza. Argumenti koje ova funkcija prima su ulazni niz, položaj od kojeg počinje traženi podniz, i duljina podniza (opcionalno). Funkcija vraća dobiveni podniz.

Primjer njenog korištenja:

```
<?php
  $niz = "O miševima i ljudima";

  echo substr($niz, 11);
  echo "<br />";
  echo substr($niz, 11, 7);
?>
```

```
i ljudima
i ljudi
```

Kao položaj zadaje se broj znakova od početka niza (brojanje znakova počinje od 0). Ako se ne navede treći argument (duljina podniza), vraća se ostatak znakova do kraja niza.

Funkcija ***str_replace*** koristi se za zamjenu dijelova niza u ulaznom nizu. Argumenti koje prima traženi su podniz, zatim podniz kojim ga treba zamijeniti, i ulazni niz. Kao četvrti (opcionalni) argument može se predati broj zamjena (koliko puta treba obaviti zamjenu). Funkcija vraća niz nastao zamjenom. Primjer korištenja ove funkcije:

```
<?php
  $niz = "O miševima i ljudima";
  $noviNiz = str_replace("ljudima", "mačkama", $niz);
  echo $noviNiz;
?>
```

```
O miševima i mačkama
```

Funkcija ***explode*** služi za pretvaranje znakovnog niza u polje. Argumenti koje ova funkcija prima su niz po kojem se ulazni niz rastavlja, zatim ulazni niz i, kao opcionalni argument, maksimalni broj članova na koje se niz može rastaviti. Funkcija vraća dobiveno polje. Primjer korištenja ove funkcije:

```
<?php
  $niz = "O miševima i ljudima";
  $polje = explode(" ", $niz);

  foreach ($polje as $clan)
  {
    echo $clan . "<br />";
  }
?>
```

```
O
miševima
i
ljudima
```

Kao niz koji služi za rastavljanje ulaznog niza, u gornjem primjeru koristi se niz " " koji sadrži samo razmak.

Funkcija ***implode*** ima obrnutu svrhu - služi za pretvaranje polja u znakovni niz. Argumenti koje prima su niz koji će se ubaciti između članova polja i polje koje treba pretvoriti u niz. Funkcija vraća dobiveni niz:

```
<?php
  $polje = array("O", "miševima", "i", "ljudima");
  $niz = implode(" - ", $polje);

  echo $niz;
?>
```

```
O - miševima - i - ljudima
```

Kao niz koji se koristi za sastavljanje dobivenog niza, u gornjem primjeru zadan je niz " – ".

7.2. Funkcije za rad s poljima

Funkcija ***count*** vraća broj članova polja. Primjer njenog korištenja:

```
<?php
  $gradovi = array("Zagreb", "Split", "Rijeka");
  echo count($gradovi);
?>
```

```
3
```

Funkcija ***in_array*** provjerava nalazi li se zadani član u polju. Argumenti koje prima su traženi član i polje. Ako je kao treći argument predana logička vrijednost *TRUE*, traženi član i članovi u polju će se uspoređivati i po tipu podataka. Ako je član pronađen, funkcija vraća *TRUE*, a u suprotnom *FALSE*:

```
<?php
  $gradovi = array("Zagreb", "Split", "Rijeka");

  if (in_array("Zagreb", $gradovi))
  {
    echo "Zagreb je pronađen!";
  }
?>
```

```
Zagreb je pronađen!
```

Funkcija ***array_sum*** će vratiti zbroj svih članova polja. Funkcija kao argument prima zadano polje:

```
<?php
  $brojevi = array(1,2,3,4,5);
  echo array_sum($brojevi);
?>
```

```
15
```

Funkcija ***shuffle*** će nasumično promijeniti poredak članova polja. Funkcija kao argument prima zadano polje (koje se prenosi po referenci, pa se ne vraća kao rezultat). Primjer korištenja funkcije:

```
<?php
  $brojevi = array(1,2,3,4,5);
  shuffle($brojevi);

  foreach($brojevi as $broj)
  {
    echo $broj . " ";
  }
?>
```

```
3 2 4 5 1
```

Dobiveni rezultat u gornjem primjeru bit će različit pri svakom pozivu funkcije *shuffle*.

Funkcija ***sort*** služi za sortiranje članova polja. Polje s brojčanim članovima se sortira po veličini, a polje sa znakovnim članovima po abecedi.

```
<?php
  $brojevi = array(3,2,4,5,1);
  sort($brojevi);

  foreach($brojevi as $broj)
  {
    echo $broj . " ";
  }
?>
```

```
1 2 3 4 5
```

Za sortiranje polja sa znakovnim ključevima potrebno je rabiti posebne funkcije za sortiranje, da se prilikom sortiranja ne bi prekinula veza između vrijednosti člana polja i njegovog ključa. Funkcija ***asort*** sortira polje sa znakovnim ključem po vrijednostima, a funkcija ***ksort*** po ključevima:

```
<?php
  $post_br = array ("Zagreb" => 10000,
                    "Rijeka" => 51000,
                    "Split" => 21000);
  asort($post_br);
  foreach($post_br as $grad => $broj)
  {
    echo "$broj $grad <br />";
  }

  echo "<br />";

  ksort($post_br);
  foreach($post_br as $grad => $broj)
  {
    echo "$grad $broj <br />";
  }
?>
```

```
10000 Zagreb
21000 Split
51000 Rijeka

Rijeka 51000
Split 21000
Zagreb 10000
```

U gornjem primjeru polje je najprije sortirano po vrijednosti člana (poštanski broj grada), a zatim po ključu (naziv grada).

Funkcija ***each*** dohvaća trenutni član polja. Nakon svakog njenog poziva unutrašnji pokazivač polja pomiče se za jedno mjesto te se tako uzastopnim pozivanjem funkcije *each* može proći kroz cijelo polje. Funkcija *each*, kao i petlja *foreach,* vraća kopiju vrijednosti člana polja pa se korištenjem nje ne može promijeniti vrijednost člana polja. Funkcija vraća polje koje sadrži ključ i vrijednost trenutnog člana kojima se može pristupiti pomoću indeksa 0 i 1, ili ključeva "key" i "value". Primjer ispisivanja članova polja pomoću funkcije *each* i petlje *while*:

```
<?php
  $post_br = array ("Zagreb" => 10000,
                    "Rijeka" => 51000,
                    "Split" => 21000);

  while($grad = each($post_br))
  {
     echo $grad["key"] . " ";
     echo $grad["value"] . "<br />";
  }

?>
```

```
Zagreb 10000
Rijeka 51000
Split 21000
```

Unixova vremenska oznaka sprema se kao 32 bitni broj, što će 2038. godine postati premaleno za spremanje trenutnog vremena.

Petlja *while* prestat će se izvršavati kad funkcija *each* dođe do kraja polja, jer će tada funkcija vratiti vrijednost *FALSE*.

7.3. Funkcije za rad s datumima i vremenom

U jeziku PHP se za rad s datumima i vremenima koristi format poznat kao *Unixova vremenska oznaka*. Radi se o broju sekundi proteklih od početka "Unixove epohe", odnosno 1.1.1970. u 0:00:00 po GMT-u.

Funkcija ***mktime*** koristi se za stvaranje Unixove vremenske oznake. Argumenti koje funkcija prima su sat, minuta, sekunda, mjesec, dan i godina. Svi argumenti su opcionalni, a ako se izostave koriste se trenutne vrijednosti: trenutna godina, trenutni dan, trenutni mjesec, itd.

Opcionalni argumenti se prilikom pozivanja funkcije *mktime* (kao i kod ostalih funkcija s opcionalnim argumentima) izostavljaju zdesna nalijevo. Dakle, ako se navede mjesec, moraju se navesti sat, minuta i sekunda, a smiju se izostaviti dan i godina.

Evo primjera u kojem se, pomoću funkcije *mktime ispisuje Unixova vremenska oznaka* za datum 13.8.2007 i vrijeme 3:24:05:

```
<?php
  $datum = mktime(3,24,5,8,13,2007);
  echo $datum;
?>
```

```
1186968245
```

Dakle, od početka "Unixove epohe" proteklo je više od milijardu sekundi.

Funkcija ***date*** služi za pretvaranje *Unixove vremenske oznake* u željeni format. Argumenti koje prima su format i vremenska oznaka. Ako se vremenska oznaka izostavi, uzima se trenutno vrijeme. Funkcija vraća znakovni niz u željenom formatu. U sljedećoj tablici prikazane su najčešće korištene oznake formata za funkciju *date*:

Oznaka formata	Značenje
d	dan u mjesecu (s vodećom nulom)
j	dan u mjesecu (bez vodeće nule)
m	mjesec (s vodećom nulom)
n	mjesec (bez vodeće nule)
y	godina kao dvoznamenkasti broj
Y	godina kao četveroznamenkasti broj
G	sat (u 24-satnom obliku, bez vodeće nule)
H	sat (u 24-satnom obliku, s vodećom nulom)
i	minute (s vodećom nulom)
s	sekunde (s vodećom nulom)

Nekoliko primjera korištenja funkcije *date*:

```
<?php
  $datum = mktime(3,24,5,8,13,2007);

  echo date("d.n.Y.", $datum);
  echo "<br/>";
  echo date("d-m-y H:i:s", $datum);
  echo "<br/>";
  echo date("G:i", $datum);
?>
```

```
13.8.2007.
13-08-07 03:24:05
3:24
```

Funkcija ***getdate*** koristi se za dohvaćanje pojedinog podatka iz vremenske oznake. Funkcija prima vremensku oznaku kao argument, a ako se pozove bez argumenta, uzima se trenutno vrijeme. Funkcija vraća polje u kojem su, preko odgovarajućeg znakovnog ključa, dostupne sljedeće vrijednosti:

Ključ	Vrijednost
seconds	sekunde
minutes	minute
hours	sati
mday	dan u mjesecu
wday	redni broj dana u tjednu (prvim danom u tjednu smatra se nedjelja, koja ima redni broj 0)
mon	mjesec
year	godina
yday	redni broj dana u godini
weekday	ime dana u tjednu
month	ime mjeseca

Primjer korištenja funkcije *getdate*:

```
<?php
  $datum = mktime(3,24,5,8,13,2007);
  $vrijeme = getdate($datum);

  echo "{$vrijeme["mday"]}.{$vrijeme["mon"]}.";
  echo "{$vrijeme["year"]} {$vrijeme["hours"]}:";
  echo "{$vrijeme["minutes"]}:{$vrijeme["seconds"]}";
?>
```

```
13.8.2007 3:24:5
```

Funkcija ***checkdate*** koristi se za provjeru je li zadani datum ispravan. Argumenti funkcije su mjesec, datum i godina. Funkcija vraća vrijednost *TRUE* ako je datum ispravan, a u suprotnom vraća *FALSE*:

```
<?php
  if (!checkdate(13,1,2007))
  {
    echo "Datum nije ispravan!";
  }
?>
```

```
Datum nije ispravan!
```

U gornjem primjeru funkcija *checkdate* vraća vrijednost *FALSE* jer datum 1.13.2007 nije ispravan. Ovu funkciju je dobro koristiti za provjeru datuma upisanog od strane korisnika.

7.4. Matematičke funkcije

Funkcija ***round*** koristi se za zaokruživanje decimalnog broja. Argumenti koji se predaju funkciji su broj koji treba zaokružiti, i (opcionalno) broj decimalnih znamenaka na koji treba zaokružiti. Ako se drugi argument izostavi, broj se zaokružuje na najbliži cijeli broj.

Funkcija ***ceil*** koristi se za zaokruživanje decimalnog broja na prvi veći cijeli broj. Kao argument joj se predaje broj koji treba zaokružiti.

Funkcija ***floor*** koristi se za zaokruživanje decimalnog broja na prvi manji cijeli broj. Kao argument predaje joj se broj koji treba zaokružiti.

Primjer korištenja funkcija *round*, *ceil* i *floor*:

```
<?php
  $broj = 4/3;

  echo $broj . "<br />";
  echo round($broj, 2) . "<br />";
  echo round($broj) . "<br />";
  echo ceil($broj) . "<br />";
  echo floor($broj);
?>
```

```
1.3333333333333
1.33
1
2
1
```

Funkcija ***max*** vraća najveći u nizu brojeva. Kao argument može se predati polje s brojevima koje treba usporediti ili dva ili više brojeva koje treba usporediti.

Funkcija ***min*** vraća najmanji u nizu brojeva. Kao argument može se predati polje s brojevima koje treba usporediti ili dva ili više brojeva koje treba usporediti.

Primjer korištenja funkcija *min* i *max*:

```
<?php
  $polje = array(1,3,7,2,5,6);

  echo "Min: ". min($polje) . "<br />";
  echo "Max: ". max($polje);
?>
```

```
Min: 1
Max: 7
```

Funkcija ***sqrt*** računa drugi korijen iz zadanog broja. Kao argument prima broj čiji korijen treba izračunati.

Primjer korištenja funkcije *sqrt* u kojem se računa korijen iz 3:

```
<?php
  echo sqrt(3);
?>
```

```
1.7320508075689
```

Funkcija ***pow*** koristi se za računanje potencija. Argumenti koje prima su baza i eksponent u operaciji potenciranja. Primjer računanja 2 na 10-tu potenciju korištenjem funkcije *pow*:

```
<?php
  echo pow(2,10);
?>
```

```
1024
```

Funkcija ***rand*** koristi se za dobivanje slučajno odabranog broja. Kao argumente može joj se predati početak i kraj raspona iz kojeg se biraju brojevi. Primjer korištenja funkcije *rand*:

Od verzije PHP-a 4.2.0 nije potrebno koristiti funkciju *srand* za zadavanje sjemena (početnog broja) generatoru slučajnih brojeva, već se on inicijalizira automatski.

```
<?php
  for ($i = 0; $i < 5; $i++)
  {
    $broj = rand();
    echo $broj . " ";
  }
?>
```

```
25607 26932 21341 1234 14755
```

Ovaj primjer će, naravno, vraćati različite rezultate pri svakom izvršavanju.

7.5. Funkcije za prekid rada skripte

Funkcija ***exit*** koristi se za prekid rada skripte. Kao argument predaje joj se tekst poruke koja se želi ispisati kao razlog prekida ili status (broj između 0 i 255) s kojim se završava izvršavanje skripte. Ako se kao argument preda broj, neće biti ispisan. Primjer korištenja funkcije *exit*:

```
<?php
  if (TRUE)
  {
    exit ("Došlo je do neočekivane pogreške.");
  }
  echo "Ovaj tekst se nikada neće ispisati.";
?>
```

```
Došlo je do neočekivane pogreške.
```

U gornjem primjeru, izvršavanje skripte će uvijek prestati, jer će funkcija *exit* uvijek biti pozvana. Uvijek će biti ispisana poruka predana funkciji *exit*, a naredba *echo* neće se nikad izvršiti.

Drugi naziv (alias) funkcije *exit* je ***die***. Nema razlike u korištenju između funkcija *exit* i *die*.

7.6. Funkcija *isset*

Funkcija ***isset*** provjerava je li nekoj varijabli pridijeljena vrijednost. Primjer korištenja funkcije *isset*:

```
<?php
  $a = 5;
  if (isset($a))
    echo "A je postavljen";
  else
    echo "A nije postavljen";
?>
```

```
A je postavljen.
```

U gornjem primjeru, varijabli *$a* je zadana vrijednost na početku pa poziv funkcije *isset* s varijablom *$a* kao argument vraća vrijednost *TRUE*.

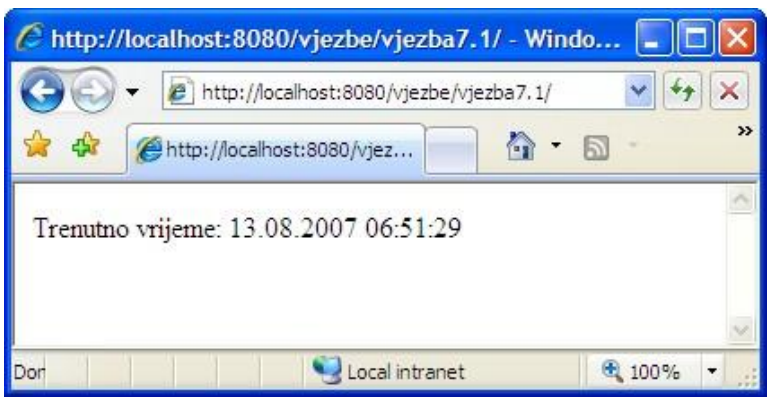
Bez prve linije kôda (*$a=5*) u gornjem primjeru bi se ispisala poruka "A nije postavljen".

Vježba 7.1 – Pretvorba iz znakovnog niza u datum

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *index.php*.
2. U datoteci spremite trenutno vrijeme u varijablu i ispišite ga:

```
<?php
  $trenutnoVrijeme = date("d.m.Y H:i:s");
  echo "Trenutno vrijeme: $trenutnoVrijeme <br/>";
?>
```

3. Spremite datoteku u mapu "..\D350\vjezbe\vjezba7.1" .
4. Pokrenite servis *EasyPHP*, ako već nije pokrenut.
5. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba7.1/* .

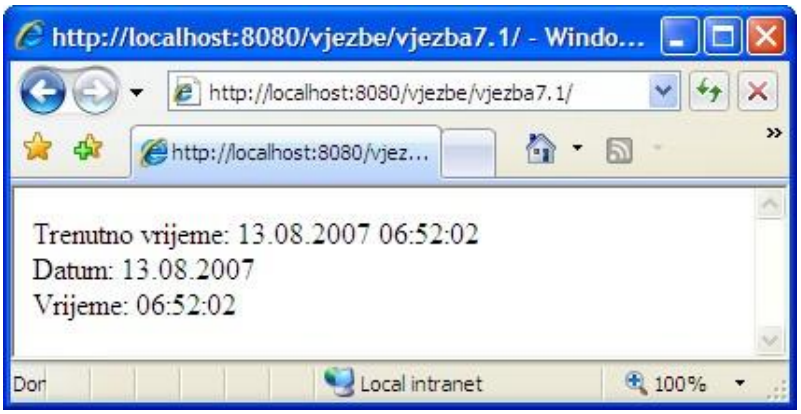


U sljedećim koracima će vrijednost iz varijable *$trenutnoVrijeme*, koja je znakovni niz, biti podijeljena na sastavne dijelove i pretvorena ponovno u vremensku oznaku.

6. Podijelite varijablu *$vrijeme* na datumski i vremenski dio pomoću funkcije *explode* i razmaka kao niza za razdvajanje:

```
  $polje = explode(" ", $trenutnoVrijeme);
  echo "Datum: $polje[0] <br/>";
  echo "Vrijeme: $polje[1] <br/>";
?>
```

Spremite datoteku i osvježite stranicu u web pregledniku.



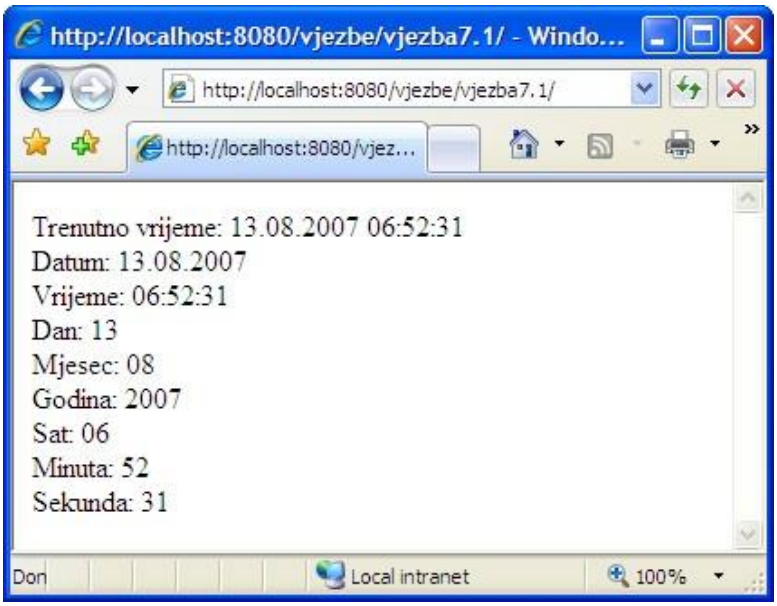
7. Podijelite dobiveni datum na dan, mjesec i godinu pomoću funkcije *explode* i točke kao niza za razdvajanje. Ispišite dobivene vrijednosti:

```
  $polje_datum = explode(".", $polje[0]);

  $dan = $polje_datum[0];
  $mjesec = $polje_datum[1];
  $godina = $polje_datum[2];

  echo "Dan: $dan <br/>";
  echo "Mjesec: $mjesec <br/>";
  echo "Godina: $godina <br/>";
?>
```

8. Podijelite dobiveno vrijeme na sat, minute i sekunde pomoću funkcije *explode* i dvotočke kao niza za razdvajanje. Ispišite dobivene vrijednosti:

```
  $polje_vrijeme = explode(":", $polje[1]);

  $sat = $polje_vrijeme[0];
  $minuta = $polje_vrijeme[1];
  $sekunda = $polje_vrijeme[2];

  echo "Sat: $sat <br/>";
  echo "Minuta: $minuta <br/>";
  echo "Sekunda: $sekunda <br/>";
?>
```

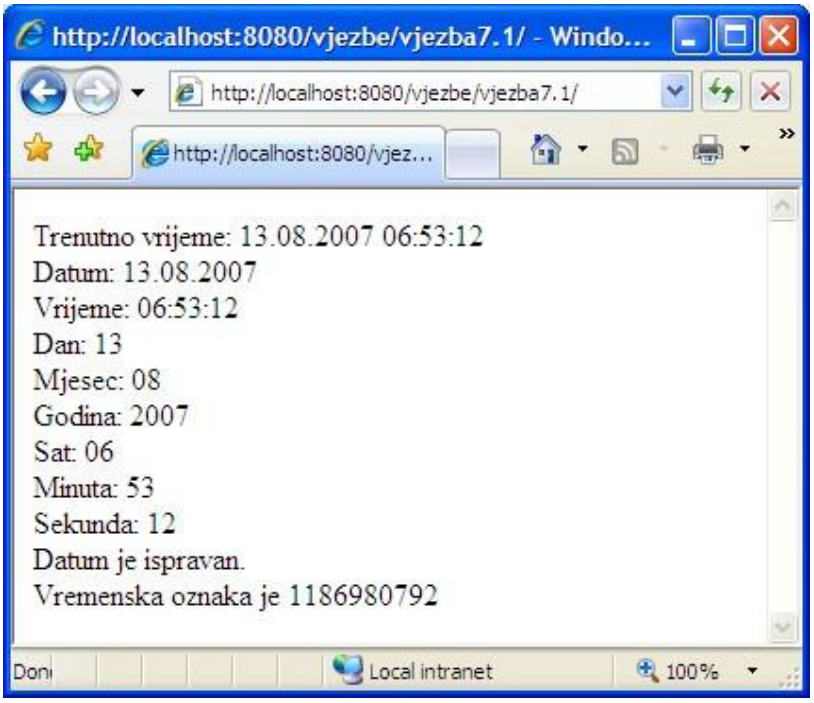
9. Spremite datoteku i osvježite stranicu u web pregledniku.



10. Provjerite je li dobiveni datum ispravan pomoću funkcije *checkdate*. Ako jest, stvorite i ispišite vremensku oznaku pomoću funkcije *mktime*:

```
  if (checkdate($mjesec,$dan,$godina))
  {
    echo "Datum je ispravan. <br/>";
    echo "Vremenska oznaka je ";
    echo mktime($sat,$minuta,$sekunda,$mjesec,$dan,$godina);
  }
?>
```

11. Spremite datoteku i osvježite stranicu u web pregledniku.



Pretvorba datuma iz tekstualnog oblika u vremensku oznaku je korisna prilikom obrade korisničkih podataka budući da korisnici unose datum kao znakovni niz.

Vježba 7.2 – Zaokruživanje prosjeka

Ova vježba je nastavak vježbe 6.3. iz prethodnog poglavlja.

1. Pokrenite program *PHP Designer* i otvorite datoteku *prosjek.php* u mapi "..\D350\vjezbe\vjezba7.2\".

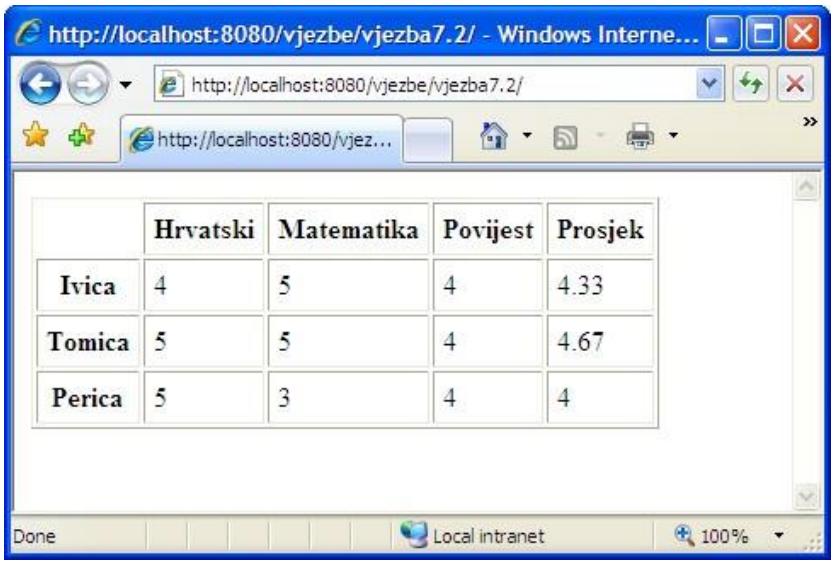
2. Promijenite funkciju *Prosjek* tako da se rezultat koji funkcija vraća zaokruži na 2 decimale korištenjem funkcije *round*:

```
<?php
  function Prosjek ($polje)
  {
    $zbroj = 0;
    $i = 0;
    foreach ($polje as $broj)
    {
      $zbroj += $broj;
      $i++;
    }
    $prosjek = $zbroj / $i;
    return round($prosjek, 2);
  }
?>
```

12. Spremite datoteku.

13. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

14. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba7.2/*



	Hrvatski	Matematika	Povijest	Prosjek
Ivica	4	5	4	4.33
Tomica	5	5	4	4.67
Perica	5	3	4	4

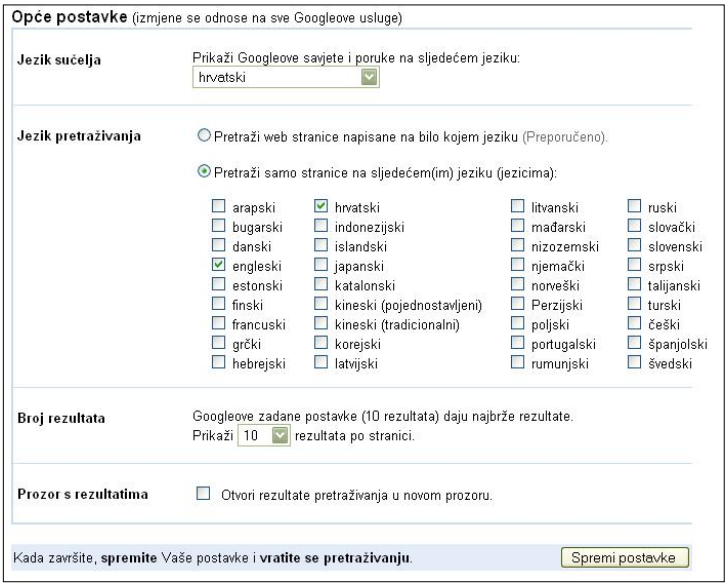
Izračunati prosjeci su isti kao i u početnoj vježbi, ali su sada prikazani tako da su zaokruženi na dvije decimale, što je preglednije.

U ovom je poglavlju obrađeno:

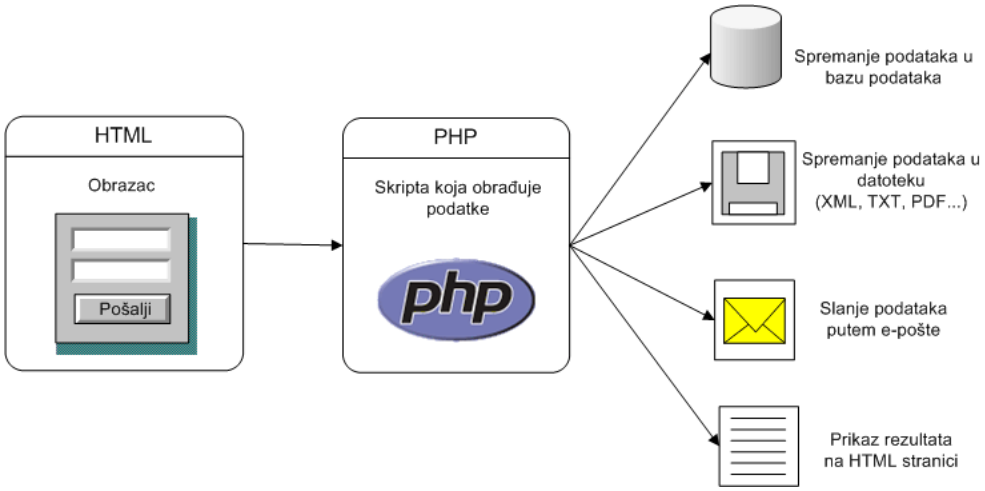
- funkcije za rad sa znakovnim nizovima – *trim*, *strtoupper*, *strtolower*, *strlen*, *substr*, *str_replace*, *explode*, *implode*
- funkcije za rad s poljima – *count*, *in_array*, *array_sum*, *shuffle*, *sort*, *asort*, *ksort*, *each*
- funkcije za rad s datumima – *mktime*, *date*, *getdate*, *checkdate*
- matematičke funkcije
- funkcije za prekid rada skripte – *exit*, *die*

8. Obrasci i prijenos podataka između skripti

Na internetu se, kao glavni način za prikupljanje podataka od strane korisnika, koriste obrasci. Obrazac na Internetu izgledom je nalik obrascu na papiru, s tim da osim polja za unos teksta, može sadržavati i interaktivne elemente: padajući izbornik, gumb za odabir. Slika ispod prikazuje primjer obrasca (podešavanje postavki na tražilici *Google*).



Učitavanjem stranice koja sadrži obrazac, korisnikov preglednik dobiva HTML kôd na temelju kojeg mu prikazuje obrazac. Nakon što korisnik popuni obrazac i klikne na dugme „Pošalji“, podaci se šalju na poslužitelj, gdje će ih dohvatiti i obraditi PHP skripta.



Postoji nekoliko mogućnosti za obradu unesenih podataka. Skripta može spremiti podatke u bazu podataka ili u datoteku na poslužitelju, ili ih poslati unutar poruke e-pošte. Uneseni podaci mogu poslužiti i za neki izračun pa se rezultat može samo prikazati korisniku bez spremanja unesenih podataka (npr. *online* kalkulator.)

8.1. Elementi obrasca

Na istoj HTML stranici moguće je imati više obrazaca. Svaki obrazac imat će vlastitu oznaku *form* i svoje elemente unutar njega.

Elementi obrasca, kao što su polje za unos teksta, padajući izbornik, i drugi, zajedno s dugmetom za slanje podataka, definiraju se pomoću HTML oznaka *input*, *textarea* i *select*.

Ove oznake potrebno je postaviti unutar HTML oznake *form* koja definira sam obrazac. Atributi koje je potrebno navesti za oznaku *form* su:

- *method* - način slanja podataka
- *action* - naziv, odnosno putanja do skripte koja će obraditi poslane podatke
- *enctype* – način kodiranja poslanih podataka (potrebno je navesti ukoliko se pomoću obrasca šalje datoteka, inače se može izostaviti)

Primjer oznake *form* mogao bi izgledati ovako:

```
<form method="get" action="Skripte/SpremiPodatke.php"
 enctype="text/plain">

   <!-- ovdje dolaze elementi obrasca -->

</form>
```

Većina elemenata obrasca ostvarena je pomoću HTML oznake *input*, a vrijednost atributa *type* određuje o kojem se zapravo elementu radi.

Polje za unos teksta


```
<input type="text" name="imeOsobe" maxlength="50"
size="30" />
```

Polje za unos teksta ostvareno je pomoću oznake *input* kojoj je vrijednost atributa *type* postavljena na *text*. Opcionalni atributi su:

Svaki element obrasca mora imati atribut *name*. Preko vrijednosti tog atributa kasnije se dohvaća vrijednost unesena u obrazac (u PHP skripti koja obrađuje podatke).

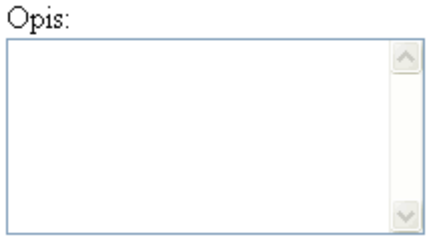
- *maxlength* - maksimalni broj znakova koji se može upisati u polje
- *size* - vizualna širina polja

Polje za unos lozinke


```
<input type="password" name="lozinka" maxlength="50"
size="30" />
```

Polje za unos lozinke ima atribut *type* postavljen na *password*. Također može imati atribute *maxlength* i *size*.

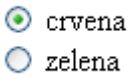
Višelinijsko polje za unos teksta




```
<textarea name="adresa" cols="40" rows="5">
</textarea>
```

Polje za unos više linija teksta ostvareno je pomoću HTML oznake *textarea*. Kod njega se ne može zadati maksimalna duljina pomoću atributa *maxlength*. Vizualna svojstva poput visine i širine polja mogu se odrediti pomoću atributa *rows* i *cols*.

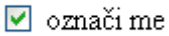
Dugme za odabir




```
<input type="radio" name="boja"  value="crvena"
 checked="checked" />

<input type="radio" name="boja"  value="zelena" />
```

Dugme za odabir ostvareno je pomoću oznake *input* kojoj je vrijednost atributa *type* postavljena na *radio*. Dugmad za odabir obično dolazi u grupama. Da bi bila povezana tako da samo jedno dugme može istovremeno biti pritisnuto, sva dugmad mora imati istu vrijednost atributa *name*. Također, da bi skripta koja prima podatke znala koje je dugme pritisnuto, svako dugme mora imati vlastitu vrijednost atributa *value*. Ako se atribut *checked* postavi na vrijednost *checked*, dugme će biti unaprijed odabrano.

Kućica za označavanje kvačicom




```
<input type="checkbox" name="oznaciMe"
 checked="checked" />
```

Ovaj element najčešće omogućuje odabir (isključenje/uključenje) neke mogućnosti, ili unos podatka o tome da li je neki uvjet istinit ili ne.

Kućica za označavanje kvačicom može biti unaprijed označena, ako se atribut *checked* postavi na vrijednost *checked*.

Polje za odabir datoteke


```
<input type="file" name="datoteka" size="30" />
```

Ovaj element obrasca omogućava, klikom na dugme „Browse..“, odabir datoteke koja će se poslati na server zajedno s obrascem. Tekst „Browse…“ ne može se promijeniti. Širina elementa može se namjestiti pomoću atributa *size*.

Padajući izbornik


```
<select name="boja">
     <option value="ff0000"  selected="selected">
          Crvena
     </option>
     <option value="00ffff">
          Zelena
     </option>
</select>
```

Kod padajućeg izbornika, pojedine vrijednosti nalaze se unutar HTML oznaka *option*. Ako želimo da poslani podatak bude drugačiji od teksta prikazanog u izborniku, oznaci *option* možemo postaviti željenu vrijednost atributa *value*. Ako želimo da neka vrijednost bude unaprijed odabrana, bit će potrebno postaviti vrijednost atributa *selected* na vrijednost *selected*.

Izbornik koji omogućuje odabir više elemenata


```
<select name="boja" multiple="multiple">
      <option value="ff0000">Crvena</option>
      <option value="00ff00">Zelena</option>
     <option value="ffff00">Žuta</option>
</select>
```

Kod ovog elementa moguće je odabrati više od jedne ponuđene vrijednosti, korištenjem tipke CTRL prilikom odabira. Ovaj element se dobiva tako da se oznaci *select* postavi atribut *multiple* na vrijednost *multiple*, a pojedine vrijednosti se također upisuju unutar oznaka *option*.

Dugme


```
<input type="submit" value="Pošalji" />

<input type="reset" value="Počisti" />

<input type="button" value="Povratak" />
```

Svaki obrazac na dnu ima dugme za slanje podataka (dugme tipa *submit*). Pritiskom na njega, upisani podaci se šalju skripti na serveru koja će ih obraditi. Ovo dugme je predodabrano dugme na obrascu. Ako pritisnemo <ENTER>, učinak će biti isti kao da smo mišem kliknuli na to dugme. Obrazac ne može imati dva dugmeta tipa *submit*.

Dugme tipa *reset* briše sve podatke upisane u obrazac, bez slanja podataka skripti na serveru.

Dugme tipa *button* nema nikakvu predodređenu funkcionalnost, već je njegovo ponašanje potrebno posebno isprogramirati (na primjer, povratak na prethodnu stranicu.)

Dugmetu tipa *submit*, *reset* ili *button* vrijednost atributa *value* označava tekst koji će pisati na dugmetu.

Slikovno dugme


```
<input type="image" src="slike/dugme.gif" />
```

Ovaj element služi kao zamjena za dugme za slanje podataka. Umjesto dugmeta, prikazuje se slika čija je lokacija navedena u *src* atributu. Klikom na sliku, podaci uneseni u obrazac šalju se skripti koja će ih obraditi.

Skriveno polje za slanje podataka

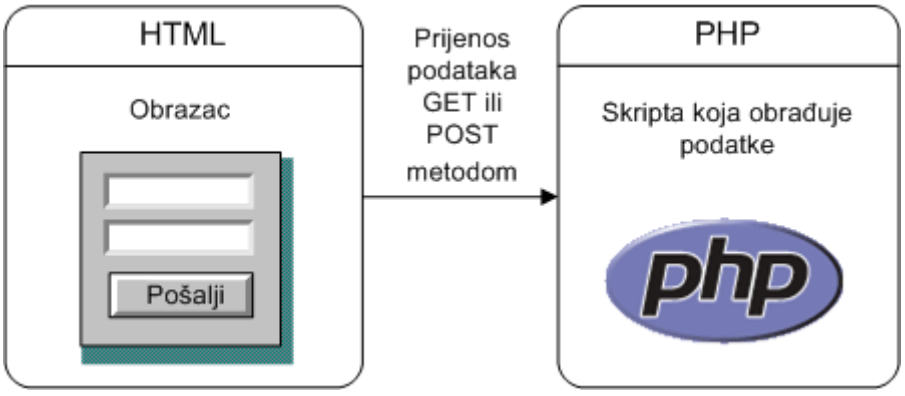
```
<input type="hidden" name="skrivenoPolje" />
```

Pomoću ovog elementa moguće je skripti koja obrađuje podatke poslati neki podatak koji nije upisao korisnik (već je stvoren u kôdu ili pročitan iz baze podataka), i to bez korisnikovog znanja. Preglednik ne prikazuje ovaj element.

8.2. Metode slanja podataka

Podaci uneseni u obrazac šalju se na poslužitelj pritiskom na dugme za slanje podataka (dugme tipa *submit*). PHP skripta kojoj se podaci šalju navedena je kao vrijednost atributa *action*, dok je način ili metoda prijenosa podataka određena atributom *method*.

Postoje dvije metode slanja podataka koje su dio HTTP protokola – metoda GET i metoda POST.



Metoda GET

Kod metode GET, podaci se šalju unutar URL adrese. Nedostatak je što su podaci tada vidljivi u navigacijskoj traci preglednika, a i dužina URL adrese je ograničena pa se veća količina podataka ne može poslati.

Obrazac kod kojeg se koristi GET metoda imat će vrijednost atributa *method* postavljenu na *get*:

```
<form method="get" action=" SpremiPodatke.php" >

</form>
```

Osim činjenice da su svi podaci, pa tako i lozinka, vidljivi u URL adresi, kod slanja podataka metodom GET postoji i problem s ograničenjem na maksimalnu veličinu podataka koji se tako mogu poslati (4 KB). Dakle, bolje je koristiti metodu POST.

Prijenos podataka GET metodom koristi se najčešće odvojeno od obrasca kad je potrebno nekoj skripti poslati malu količinu podataka. Većinom se susreće u linkovima, kada se npr. u URL adresi skripte koja prikazuje vijest proslijedi identifikator vijesti u bazi. Takva URL adresa izgledala bi ovako:

ImeSkripte?nazivParametra=vrijednost

Primjer:

`index.php?id=28`

Ako se u URL adresi prosljeđuje više od jedne vrijednosti, koristi se znak & za dodavanje novog parametra:

ImeSkripte?naziv1=vrijednost1&naziv2=vrijednost2

Primjer:

`index.php?id=28&backPID=28`

Metoda POST

Kod metode POST, podaci se šalju unutar tijela HTTP zahtjeva i ne postoji ograničenje na količinu podataka koja se može poslati. Obrazac koji šalje podatke metodom POST, mora imati vrijednost atributa *method* postavljenu na *post*:

```
<form method="post" action="SpremiPodatke.php" >

</form>
```

8.3. Dohvaćanje podataka unesenih u obrazac

PHP skripta koja obrađuje podatke (dakle skripta koja je navedena u *action* atributu) mora dohvatiti poslane podatke kako bi se s njima izvršile daljnje akcije: spremanje u bazu podataka, slanje e-poštom, izračun rezultata.

Unutar te skripte, poslani podaci bit će dostupni u polju *$_GET* ili u polju *$_POST*, ovisno o korištenoj metodi prijenosa podataka. To su asocijativna polja čijem se članu može pristupiti preko vrijednosti atributa *name* željenog elementa obrasca.

Pretpostavimo da obrazac izgleda kao u ovom primjeru:

```
<form method="metodaSlanja" action="Skripta.php" >
    <input type="text" name="imeOsobe" />
    ...
</form>
```

U skripti koja obrađuje obrazac (*Skripta.php*) ovako će se dohvatiti vrijednost upisana u polje za unos teksta:

- za metodu GET:

```
<?php
      $imeOsobe = $_GET['imeOsobe'];
?>
```

- za metodu POST:

```
<?php
      $imeOsobe = $_POST['imeOsobe'];
?>
```

Osim polja *$_GET* i *$_POST*, postoji i polje *$_REQUEST* preko kojeg se može pristupiti podacima poslanim bilo metodom GET, bilo metodom POST. No, ono se rjeđe koristi.

Vježba 8.1 – Obrazac

1. Pokrenite program *PHP Designer* i stvorite novu HTML datoteku.
2. Spremite datoteku pod imenom "Adresar.htm" (u mapu "..\D350\vjezbe\vjezba8.1").
3. U datoteku dodajte oznake *html, head, title* i *body*.
4. Unutar oznake *title* upišite "Adresar". Dodajte oznaku *meta* za ispravan prikaz hrvatskih dijakritičkih znakova.
5. Unutar oznake *body* dodajte oznaku *h1* unutar koje upišite "Unos adrese".
6. Unutar oznake *head* dodajte oznaku *meta* za ispravan prikaz hrvatskih dijakritičkih znakova.

   Dosad napisani kôd trebao bi izgledati ovako:

```
<html>
<head>
    <title>Adresar</title>
     <meta http-equiv="Content-Type"
          content="text/html; charset=windows-1250" />
</head>
<body>
    <h1>Unos adrese</h1>
</body>
</html>
```

Radi ispravnog prikazivanja hrvatskih dijakritičkih znakova, preporučljivo je unutar *head* oznake dodati *meta* oznaku s odgovarajućom vrijednosti za *charset* (kôdnu stranicu – način na koji se znakovi pohranjuju u binarnom obliku).

Na tečaju će se koristiti kôdna stranica windows-1250, a za prikaz naših znakova koriste se još i kôdne stranice utf-8 i iso-8859-2 (latin 2).

7. Unutar oznake *body* dodajte oznaku *form.* Kao način prijenosa podataka zadajte metodu GET, a za ime skripte kojoj se podaci šalju navedite "SpremiAdresu.php".

```
<form method="get" action="SpremiAdresu.php">

</form>
```

8. Prvi element obrasca bit će ime osobe koja se unosi u adresar. Unutar oznake *form* napišite „Ime:“ i dodajte oznaku *br.*
9. Dodajte oznaku *input* tipa *text*. Vrijednost atributa *name* neka joj bude "ime". Radi oblikovanja dodajte na kraju dvije oznake *br*.

```
Ime:<br/>
<input type="text" name="ime" />
<br/><br/>
```

10. Sljedeći element obrasca bit će adresa osobe. Upotrijebit ćemo oznaku *textarea* kako bi se adresa mogla unijeti u više redaka. Napišite „Adresa:“ i dodajte oznaku *br*. Dodajte oznaku *textarea* s imenom „adresa“ i poslije nje još dvije oznake *br*.

```
Adresa:<br/>
<textarea name="adresa"></textarea>
<br/><br/>
```

11. Sljedeći element obrasca bit će grad. Grad će se odabirati pomoću padajućeg izbornika (oznaka *select*). Pojedini gradovi koji će biti ponuđeni na odabir nalazit će se unutar oznaka *option*. Dodajte sljedeći kôd:

```
Grad:<br/>
<select name="grad">
      <option>Zagreb</option>
      <option>Split</option>
</select>
<br/><br/>
```

12. Zatim, pomoću dugmeta za odabir bit će potrebno odabrati spol osobe. Potrebno je dodati dvije oznake *input* tipa *radio*, čiji atribut *name* mora biti jednak. Svakoj oznaci *input* treba postaviti atribut *value* na vrijednost koja označava napravljeni odabir.

```
Spol:<br/>
<input type="radio" value="M" name="spol" /> muški<br/>
<input type="radio" value="Ž" name="spol" /> ženski
<br/><br/>
```

13. Sljedeći element obrasca je okvir za potvrdu pomoću kojeg će vlasnik adresara moći označiti je li mu unesena osoba dobar prijatelj. Treba dodati oznaku *input* tipa *checkbox*. Atribut checked možemo postaviti ako želimo da okvir bude unaprijed potvrđen (tj.označen kvačicom).

```
Prijatelj:<br/>
<input type="checkbox" checked="checked"
name="prijatelj"/>
<br/><br/>
```

14. Posljednji elementi ovog obrasca su dugme za slanje obrasca i dugme za čišćenje obrasca. Dodajte dvije oznake *input*, tipova *submit* i *reset*. Prvom dugmetu kao vrijednost value upišite „Spremi“ a drugom „Odustani“.

```
<input type="submit" value="Spremi" />
<input type="reset" value="Odustani" />
```

15. Spremite datoteku pod imenom "UnosAdrese.htm" (u mapu "..\D350\vjezbe\vjezba8.1").

16. Pokrenite servis *EasyPHP* .

17. Pokrenite vaš web preglednik i upišite adresu *http://localhost/vjezbe/vjezba8.1/UnosAdrese.htm* .

    Dobivena stranica trebala bi izgledati kao na slici ispod.

18. Vratite se u program *PHP Designer* i stvorite novu PHP datoteku. Ona će služiti za obradu podataka unesenih u obrazac.

19. U datoteku dodajte oznake *html, head, title, meta, body i h2,* slično kao i za datoteku *UnosAdrese.htm*.

```
<html>
<head>
    <title>Adresar</title>
    <meta http-equiv="Content-Type"
       content="text/html; charset=windows-1250" />
</head>
<body>
<h2>Spremljena adresa</h2>

</body>
</html>
```

20. Poslije oznake *h2* dodajte oznake za PHP kôd i unutar njih sljedeće naredbe:

```
<?php
      $ime = $_GET['ime'];
      $adresa = $_GET['adresa'];
      $grad = $_GET['grad'];
      $spol = $_GET['spol'];
      $prijatelj = $_GET['prijatelj'];
?>
```

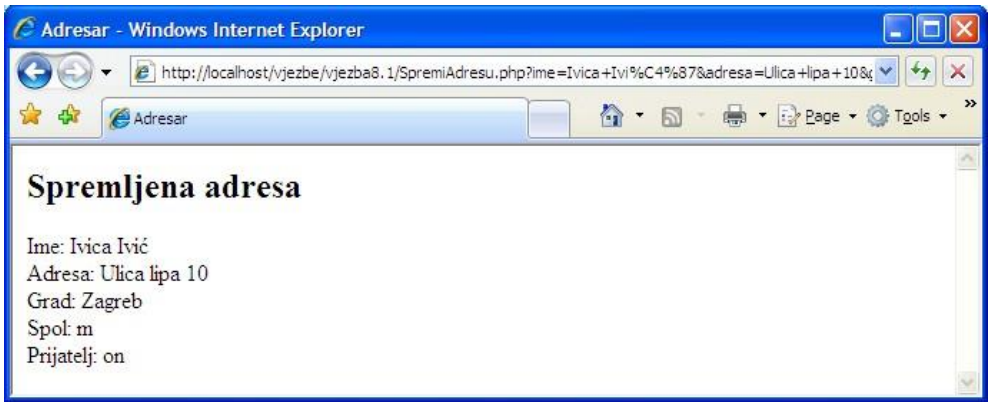
U varijable *$ime, $adresa, $grad, $spol* i *$prijatelj* spremamo podatke koje skripta dobiva putem metode GET.

21. Poslije ovih naredbi dodajte sljedeće naredbe, koje služe za ispis vrijednosti varijabli:

```
      echo "Ime: $ime <br/>";
      echo "Adresa: $adresa <br/>";
      echo "Grad: $grad <br/>";
      echo "Spol: $spol <br/>";
      echo "Prijatelj: $prijatelj <br/>";
```

22. Spremite datoteku pod imenom "SpremiAdresu.php" (u mapu "..\D350\vjezbe\vjezba8.1").

23. Vratite se u web preglednik. Ispunite obrazac podacima i pritisnite dugme „Spremi“. Rezultat bi trebao biti sličan onome na slici ispod.



Pogledajmo URL adresu u navigacijskoj traci preglednika. Preusmjereni smo na *SpremiAdresu.php*, a podaci koje smo upisali u obrazac dodani su kao parametri u URL adresu.

Varijabla *$spol* sadrži vrijednost atributa *value* za odabrano dugme za odabir (u gornjem slučaju „m“), a varijabla *$prijatelj* vrijednost „on“, koja označava da je okvir za potvrdu bio označen kvačicom (uključen). Na sljedeći način bi se poslani podaci mogli preglednije ispisati:

24. Vratite se u program *PHP Designer*. U datoteci *SpremiAdresu.php* izbrišite ove dvije naredbe:

```
$spol = $_GET['spol'];
$prijatelj = $_GET['prijatelj'];
```

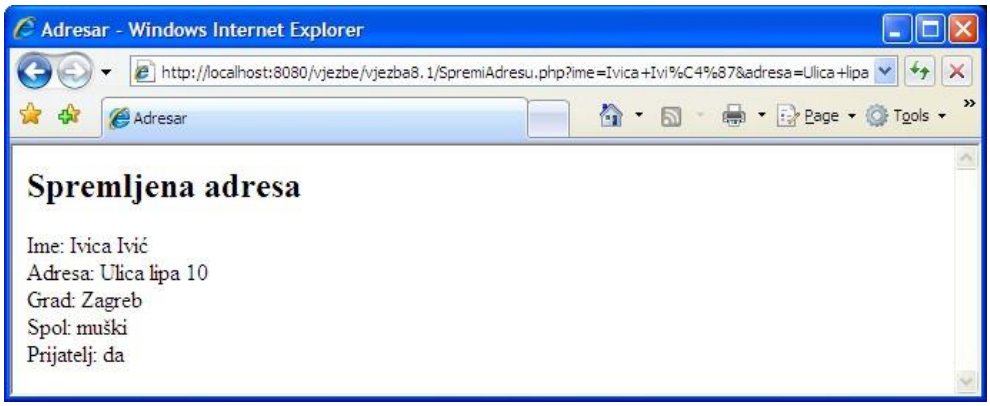
Umjesto njih napišite sljedeće naredbe:

```
if (isset($_GET['spol']))
{
  if ($_GET['spol'] == "M")
     $spol = "muški";
   else
     $spol = "ženski";
}
 else
  $spol = "";

if (isset($_GET['prijatelj']))
      $prijatelj = "da";
else
      $prijatelj = "ne";
```

Funkcija *isset* koristi se za provjeru je li postavljena vrijednost *$_GET['spol']*, odnosno *$_GET['prijatelj']*. Ako kućica za označavanje ili nijedno dugme za odabir nisu označeni, funkcija *isset* vratit će vrijednost *FALSE*.

Ako je za element obrasca *spol* poslana vrijednost „M“, varijabla *$spol* dobit će vrijednost „muški“, a inače „ženski“. Slično, ako je za element obrasca *prijatelj* poslana vrijednost „on“ (ako je okvir za potvrdu bio označen), varijabla *$prijatelj* dobit će vrijednost „da“, a inače „ne“.

25. Spremite izmijenjenu datoteku. U web pregledniku opet otvorite datoteku *Adresar.htm*. Popunite obrazac i pritisnite „Spremi“. Rezultat bi trebao biti sličan onome na slici ispod.



Vježba 8.2 – Slanje podataka metodom *POST*

1. U programu *PHP Designer* otvorite datoteku *Adresar.htm* iz prethodne vježbe. Oznaci *form* promijenite vrijednost atributa *method* iz *get* u *post*. Spremite izmjene.

2. Otvorite i datoteku *SpremiAdresu.php* iz prethodne vježbe. Napravite sljedeće izmjene i spremite ih:
   - promijenite *$_GET['ime']* u *$_POST['ime']*
   - promijenite *$_GET['adresa']* u *$_POST['adresa']*
   - promijenite *$_GET['grad']* u *$_POST['grad']*
   - promijenite *$_GET['spol']* u *$_POST['spol']*
   - promijenite *$_GET['prijatelj']* u *$_POST['prijatelj']*

3. U web pregledniku otvorite datoteku *Adresar.htm*. Popunite obrazac i pritisnite dugme „Spremi“.

   Rezultat će biti isti kao i u prethodnoj vježbi. Jedina razlika je u tome što poslani podaci više nisu vidljivi u URL adresi.

U ovom je poglavlju obrađeno:

- obrazac i njegovi elementi
- slanje podataka metodama GET i POST
- pristupanje poslanim podacima

9. Rad s datotekama

PHP ima brojne mogućnosti za rad s datotekama.

Slanje datoteke na poslužitelj (upload) u PHP-u je vrlo jednostavno, a omogućeno je pomoću elementa obrasca *input* čiji je atribut *type* postavljen na "file".

9.1. Otvaranje i zatvaranje datoteke

Da bi se s datotekom obavila neka akcija, najprije ju je potrebno otvoriti. U tu svrhu služi funkcija ***fopen***. Njen poziv izgleda ovako:

```
$datoteka = fopen ("vjezbe/datoteka.txt","r");
```

Funkcija *fopen* može otvoriti i datoteke koje nisu na istom poslužitelju, preko HTTP ili FTP protokola. Kao putanju potrebno je navesti URL adresu do datoteke, npr. "http://www.srce.hr/datoteka.xml" Naravno, potrebno je da datoteka bude javno dostupna.

Funkcija *fopen* vraća pokazivač na datoteku (*file pointer*, odnosno *file handle*). Taj pokazivač (u gornjem primjeru varijabla *$datoteka*) je varijabla tipa *resource*, i sav daljnji rad s datotekom se odvija pomoću njega.

Prvi argument koji funkcija *fopen* prima je putanja do datoteke. Putanja do datoteke može biti apsolutna (potpuna putanja do datoteke, npr. "C:/www/D350/vjezbe/datoteka.txt" ili relativna (putanja u odnosu na mapu gdje se nalazi skripta, npr. "vjezbe/datoteka.txt"). Da bi skripte bez problema mogle raditi na različitim poslužiteljima, bolje je upotrebljavati relativne putanje.

Relativne putanje se promatraju u odnosu na trenutnu mapu (mapu gdje se nalazi skripta iz koje se poziva funkcija *fopen*.)

Ako se datoteka koju se želi otvoriti nalazi u trenutnoj mapi, dovoljno je navesti ime datoteke, npr. "datoteka.txt".

Ako se datoteka nalazi u podmapi trenutne mape, putanja do nje ima oblik "podmapa/datoteka.txt".

Ako se datoteka nalazi u mapi koja je iznad trenutne mape, relativna putanja ima oblik "../datoteka.txt" .

Za navođenje putanja koristi se kosa crta ("/"). Na Windows poslužiteljima, za navođenje apsolutne putanje može se, ali i ne mora, koristiti i obrnuta kosa crta ("\").

Drugi argument funkcije je način rada s datotekom. Radi se o tekstualnoj konstanti koja opisuje koje će se akcije izvršavati nad datotekom. U gornjem primjeru koristi se način rada za čitanje ("r"), a svi načini rada su dani u sljedećoj tablici:

Način rada		Objašnjenje
r	(read)	čitanje iz datoteke
r+		čitanje i pisanje
w	(write)	pisanje (ako datoteka ne postoji, stvara se; postojeći sadržaj se briše)
w+		pisanje i čitanje (ako datoteka ne postoji, stvara se; postojeći sadržaj se briše)
a	(append)	dodavanje u datoteku (postojeći sadržaj se ne briše)
a+		dodavanje i čitanje (postojeći sadržaj se ne briše)
b	(binary)	rad s binarnom datotekom; koristi se u kombinaciji s ostalim načinima (rb – čitanje iz binarne datoteke, wb+ - pisanje i čitanje u binarnu datoteku); potrebno rabiti samo na Windowsovim poslužiteljima

Preporuča se navođenje najjednostavnijeg potrebnog načina rada (dakle, ako se planira samo čitati iz datoteke, bolje je navesti "r" nego "r+"), jer se time štede resursi poslužitelja.

Nakon poziva funkcije *fopen*, potrebno je provjeriti je li otvaranje datoteke uspjelo. Pogreška se može dogoditi u slučaju da datoteka ne postoji, putanja nije točno navedena, ili ako je dozvoljeno samo čitanje iz datoteke, a pokušava je se otvoriti u načinu rada za pisanje. Provjera je li otvaranje datoteke uspjelo svodi se na provjeru je li vraćeni pokazivač prazan ili ne. Rad s datotekom treba nastaviti samo ako je datoteka uspješno otvorena.

```
if ($datoteka)
{
   // rad s datotekom
}
```

Kad je rad s datotekom završen, potrebno ju je zatvoriti (opet radi štednje resursa poslužitelja). Zatvaranje datoteke obavlja se pomoću funkcije ***fclose***. Ovoj se funkciji kao argument predaje pokazivač na datoteku, a primjer njenog poziva izgleda ovako:

```
fclose ($datoteka);
```

Koraci koje je potrebno obaviti prilikom rada s datotekom su:

- otvaranje datoteke
- provjera je li datoteka uspješno otvorena
- čitanje iz datoteke ili pisanje u datoteku
- zatvaranje datoteke.

Primjer u kojem se obavljaju svi potrebni koraci:

```
// otvaranje datoteke
$datoteka = fopen("Datoteka.txt", "r");

// provjera je li datoteka uspješno otvorena
if ($datoteka)
{
   // rad s datotekom

   // zatvaranje datoteke
   fclose($datoteka);
}
```

9.2. Čitanje iz datoteke

Za čitanje iz datoteke može se koristiti funkcija ***fread***.

```
$procitano = fread($datoteka, $duljina);
```

Funkcija *fread* kao argumente prima pokazivač na datoteku i duljinu niza koji se čita iz datoteke (u bajtovima).

Funkcija ***fgets*** omogućuje čitanje jednog retka iz datoteke:

```
$redak = fgets($datoteka);
```

Funkcija *fgets* prima pokazivač na datoteku a vraća pročitani redak.

Za čitanje jednog znaka iz datoteke koristi se funkcija ***fgetc***.

```
$znak = fgetc($datoteka);
```

Funkcija *fgetc* prima pokazivač na datoteku a vraća pročitani znak.

Pokazivač na datoteku pamti trenutnu poziciju u datoteci koja se čitanjem pomoću funkcija *fread*, *fgets* i *fgetc* pomiče za zadani broj bajtova, jedan redak, odnosno jedan znak.

Provjera je li datoteka došla do kraja obavlja se pomoću funkcije ***feof***. Ova funkcija prima pokazivač na datoteku a vraća vrijednost *true* ako je pokazivač datoteke na kraju datoteke.

Ovako bi izgledalo čitanje cijele datoteke redak po redak:

```
while (!feof($datoteka))
{
   $redak = fgets($datoteka);
}
```

Unutar *while* petlje izvršava se čitanje jednog retka iz datoteke. Nakon što se pročita posljednji redak, funkcija *feof* će vratiti *false* i petlja će se prestati izvršavati.

9.3. Pisanje u datoteku

Za pisanje u datoteku koristi se funkcija ***fwrite***.

```
fwrite($datoteka, $tekst, $duljina);
```

Ova funkcija prima tri argumenta: pokazivač na datoteku, tekst koji treba upisati i duljinu teksta kojeg treba upisati (u bajtovima). Duljina je opcionalni argument – ako se ne navede, u datoteku se zapisuje sav predani tekst, a ako se navede, zapisuje se prvih toliko bajtova teksta.

Zapis nekog niza znakova u datoteku izgledat će ovako:

```
fwrite($datoteka, "Ovo je niz znakova.");
```

Ako se niz znakova želi u datoteku zapisati kao redak, potrebno mu je na kraj dodati znak za novi redak ("\n"):

```
fwrite($datoteka, "Ovo je redak.\n");
```

9.4. Slanje datoteke na poslužitelj

Datoteka se može poslati s korisnikovog na poslužiteljsko računalo pomoću obrasca. Za to služi polje za odabir datoteke, odnosno element obrasca *input*, čiji je atribut *type* postavljen na vrijednost *file*. Također, samom je obrascu potrebno postaviti atribut *enctype* na vrijednost "multipart/form-data", a za slanje podataka se mora koristiti metoda POST.

Klikom na dugme "Browse" otvara se samo dijaloški okvir za odabir datoteke na klijentskom računalu. Nakon što se datoteka odabere i klikne "OK", datoteka još nije poslana. Slanje datoteke na poslužitelj događa se tek kad se klikne na dugme obrasca za slanje podataka (dugme tipa *submit*).

```
<form method="POST" action="Skripta.php"
 enctype="multipart/form-data">

   <input type="file" name="datoteka"/>

</form>
```

Na poslužiteljskoj strani, poslanoj datoteci i informacijama o njoj može se pristupiti preko polja *$_FILES,* slično kao što se drugim podacima upisanim u obrazac može pristupiti preko polja *$_GET*, odnosno *$_POST*.

Polje *$_FILES* je dvodimenzionalno polje, u kojem prva dimenzija označava naziv polja za odabir (atribut *name*) budući da ih može biti više unutar istog obrasca. Druga dimenzija se odnosi na pojedinu vrijednost vezanu uz datoteku. Polje *$_FILES* (ako je naziv polja za odabir "datoteka") ima ove elemente:

Element	Objašnjenje
$_FILES["datoteka"]["name"]	naziv datoteke na korisnikovu računalu
$_FILES["datoteka"]["type"]	tip datoteke (u MIME formatu)
$_FILES["datoteka"]["size"]	veličina datoteke
$_FILES["datoteka"]["tmp_name"]	privremeni naziv (i putanja do) datoteke na poslužitelju
$_FILES["datoteka"]["error"]	kôd greške (ako je došlo do greške); 0 ako je sve u redu

Koristeći se poljem *$_FILES*, ovako bi se u skripti koja obrađuje poslane podatke dohvatili naziv i veličina datoteke:

```
$imeDatoteke = $_FILES["datoteka"]["name"];
$velicina = $_FILES["datoteka"]["size"];
```

Provjera je li korisnik u obrascu uopće odabrao datoteku, može se izvršiti provjerom postoji li u polju *$FILES* element za odgovarajuće polje za odabir. Ako ne postoji element *$FILES["datoteka"]["name"]*, datoteka nije poslana.

```
if ($_FILES["datoteka"]["name"])
{
   // spremi poslanu datoteku
}
```

Također, uputno je provjeriti je li se dogodila greška prilikom slanja pa će potpunija provjera izgledati ovako:

```
if ($_FILES["datoteka"]["name"] &&
    $_FILES["datoteka"]["size"] != 0)
{
   // spremi poslanu datoteku
}
```

U gornjem primjeru provjerava se je li veličina poslane datoteke različita od nule. Ova provjera pouzdanija je od provjere vrijednosti *$_FILES["datoteka"]["error"]* koja ne radi ispravno u nekim verzijama PHP-a, a može ovisiti i o postavkama poslužitelja.

Poslana datoteka spremljena je na privremeno mjesto čija se putanja može dobiti pomoću vrijednosti *$_FILES["datoteka"]["tmp_name"]*. Da bi se poslana datoteka spremila u željenu mapu (i pod željenim imenom), potrebno je koristiti se funkcijom ***move_uploaded_file*** čiji je primjer korištenja dan u nastavku:

```
   // spremi poslanu datoteku
   move_uploaded_file($_FILES["datoteka"]["tmp_name"],
    "datoteka.txt");
```

Prvi argument funkcije je trenutna, privremena putanja do datoteke, a drugi je željena putanja.

Vježba 9.1 – Zapisivanje u datoteku

Ova vježba se nastavlja na vježbu 8.2. iz prethodnog poglavlja.

1. Pokrenite program *PHP Designer.* U mapi "…\D350\vjezbe\vjezba9.1" otvorite datoteku *SpremiAdresu.php*. Primjećujete da se u toj mapi nalaze samo datoteke *SpremiAdresu.php* i *UnosAdrese.htm*.

2. U datoteku *SpremiAdresu.php*, prije završne oznake za PHP kôd ("?>"), dodajte naredbu za otvaranje datoteke u načinu rada za dodavanje.

```
$datoteka = fopen("Adresar.txt", "a");
```

   Ako datoteka *Adresar.txt* ne postoji, bit će stvorena, a ako postoji, sadržaj će biti dodan na kraj. Pokazivač na datoteku bit će spremljen u varijablu *$datoteka*.

3. Nakon te naredbe dodajte naredbu za zapisivanje sadržaja u datoteku (možete je napisati u jednom redu, ovdje je prelomljena iz grafičkih razloga):

```
fwrite($datoteka,
 "\n$ime\n$adresa\n$grad\n$spol\n$prijatelj");
```

   U datoteku otvorenu pomoću prve naredbe zapisat će se vrijednosti varijabli *$ime*, *$adresa*, *$grad*, *$spol* i *$prijatelj*. Vrijednosti će biti razdvojene pomoću znaka za novi red ("\n").

4. Nakon ove naredbe, dodajte naredbu za zatvaranje datoteke.

```
fclose($datoteka);
```

5. Spremite datoteku.

6. Pokrenite servis *EasyPHP* (ako već nije pokrenut).

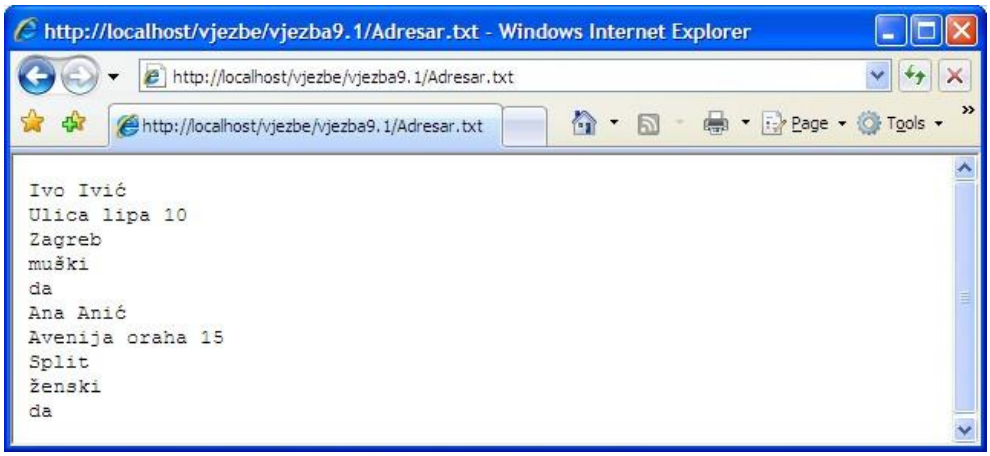
7. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba9.1/UnosAdrese.htm* .

8. Popunite obrazac i kliknite na "Spremi".

9. Otvorite mapu "…\D350\vjezbe\vjezba9.1". Primjećujete da je sada u njoj stvorena datoteka *Adresar.txt*.

10. Vratite se natrag na unos adrese. Popunite obrazac novim podacima i kliknite na "Spremi".

11. Otvorite datoteku *Adresar.txt* u web pregledniku tako da upišete adresu *http://localhost/vjezbe/vjezba9.1/Adresar.txt).* Podaci upisani u obrazac bit će spremljeni u datoteci.



Vježba 9.2 – Čitanje iz datoteke

1. Pokrenite program *PHP Designer.* U mapi "…\D350\vjezbe\vjezba9.2", stvorite novu datoteku *PregledAdresa.php*.

2. U datoteku upišite sljedeći kôd:

Ova vježba se nastavlja na prethodnu vježbu.

```
<html>
<head>
   <title>Adresar</title>
   <meta http-equiv="Content-Type" content="text/html;
    windows-1250">
</head>
<body>
   <h2>Pregled adresa</h2>
   <table border="1">
      <tr>
         <th>Ime</th>
         <th>Adresa</th>
         <th>Grad</th>
         <th>Spol</th>
         <th>Prijatelj</th>
      </tr>
      <?php

      ?>
   </table>
   <br/>
   <a href="UnosAdrese.htm">Unos nove adrese</a>
</body>
</html>
```

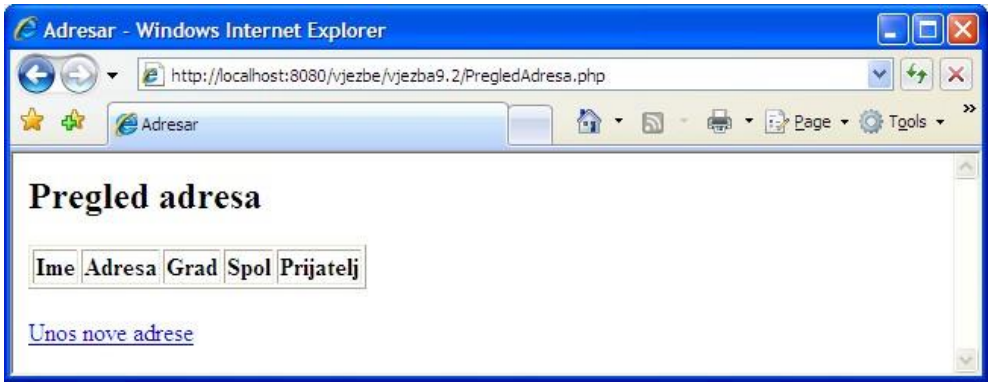
3. Unutar oznake *head* nalazi se, osim naslova stranice, i oznaka *meta* koja osigurava ispravan prikaz teksta u pregledniku. Unutar oznake *body* imamo oznaku *h2* (naslov veličine 2), tablicu, oznaku *br* (radi estetike) i link na stranicu *UnosAdrese.htm*.

Unutar tablice se nalazi redak sa zaglavljem (oznake *th* služe za ćelije zaglavlja tablice.) Ostali reci tablice bit će ispisani pomoću PHP kôda, koji će doći unutar oznaka "<?php" i "?>".

4. Spremite datoteku.

5. Pokrenite servis *EasyPHP* (ako već nije pokrenut).

6. U web preglednik upišite adresu:
   http://localhost/vjezbe/vjezba9.2/PregledAdresa.php .



7. Vratite se u program PHP Designer, i u datoteku *PregledAdresa.php*, unutar oznaka za PHP kôd, dodajte sljedeću naredbu:

```
$datoteka = fopen("Adresar.txt", "r");
```

Ova će naredba otvoriti datoteku *Adresar.txt* za čitanje, a pokazivač na datoteku bit će spremljen u varijabli *$datoteka*.

8. Zatim dodajte sljedeći kôd:

```
if ($datoteka)
{
    // čitanje iz datoteke
}
```

Ovo je provjera je li datoteka uspješno otvorena. Čitanje iz datoteke ima smisla samo ako je dobiven pokazivač na datoteku, pa će sve naredbe za čitanje iz datoteke doći će unutar ovog *if* bloka.

9. Unutar *if* bloka dodajte masno otisnuti kôd:

```
if ($datoteka)
{
    // čitanje iz datoteke
    fgets($datoteka);
    while (!feof($datoteka))
    {

    }
}
```

Na početku datoteke nalazi se jedan prazan redak, i najprije je potrebno pročitati njega pomoću funkcije *fgets*. Nakon toga, unutar *while* petlje će se obavljati čitanje ostalih zapisa iz datoteke. Pomoću funkcije *feof* provjerava se je li pročitan kraj datoteke. Ako jest, petlja će se prestati izvršavati.

Unutar petlje *while* dodajte masno otisnute naredbe:

```
if ($datoteka)
{
    // čitanje iz datoteke
    fgets($datoteka);
    while (!feof($datoteka))
    {
       $ime = fgets($datoteka);
       $adresa = fgets($datoteka);
       $grad = fgets($datoteka);
       $spol = fgets($datoteka);
       $prijatelj = fgets($datoteka);
    }
}
```

U jednom se prolazu petlje *while* čita pet redaka iz datoteke, pomoću funkcije *fgets*. Prvi se redak sprema u varijablu *$ime*, drugi u varijablu *$adresa*, treći u varijablu *$grad*, četvrti u varijablu *$spol* i peti u varijablu *$prijatelj*.

10. Dodajte ove naredbe nakon naredbi dodanih u prethodnom koraku:

```
if ($datoteka)
{
    // čitanje iz datoteke
    fgets($datoteka);
    while (!feof($datoteka))
    {
       $ime = fgets($datoteka);
       $adresa = fgets($datoteka);
       $grad = fgets($datoteka);
       $spol = fgets($datoteka);
       $prijatelj = fgets($datoteka);

       echo "<tr>";
       echo "<td>$ime</td>";
       echo "<td>$adresa</td>";
       echo "<td>$grad</td>";
       echo "<td>$spol</td>";
       echo "<td>$prijatelj</td>";
       echo "</tr>";
    }
}
```

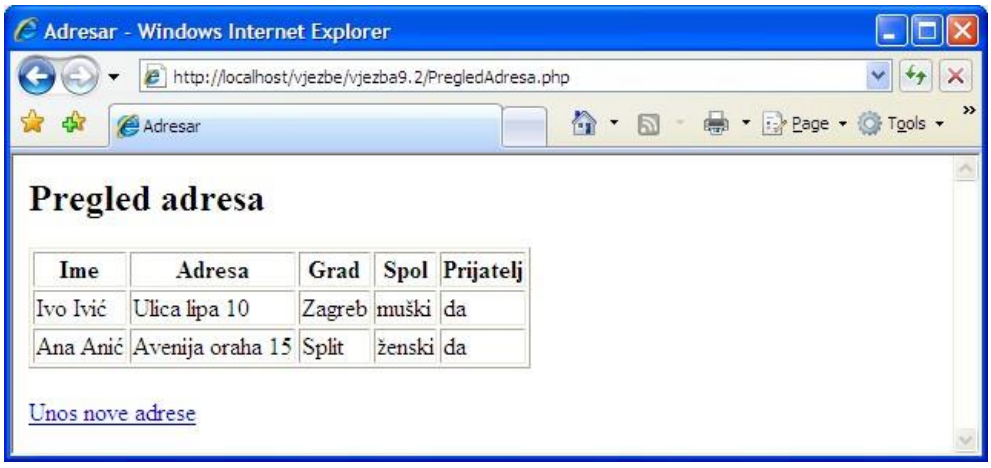
Ove naredbe ispisat će jedan redak tablice (oznaka *tr*), a vrijednosti varijabli *$ime*, *$adresa*, *$grad*, *$spol* i *$prijatelj* bit će ispisane unutar ćelija u retku (oznaka *td*). Za svaki prolaz petlje *while* ispisat će se po jedan redak tablice.

11. Poslije petlje *while* dodajte naredbu za zatvaranje datoteke:

```
if ($datoteka)
{
    fgets($datoteka);
    while (!feof($datoteka))
    {
       $ime = fgets($datoteka);
       $adresa = fgets($datoteka);
       $grad = fgets($datoteka);
       $spol = fgets($datoteka);
       $prijatelj = fgets($datoteka);

       echo "<tr>";
       echo "<td>$ime</td>";
       echo "<td>$adresa</td>";
       echo "<td>$grad</td>";
       echo "<td>$spol</td>";
       echo "<td>$prijatelj</td>";
       echo "</tr>";
    }

    fclose($datoteka);
}
```

12. Spremite datoteku.

13. Osvježite stranicu *PregledAdresa.php* u web pregledniku (pritiskom na tipku <F5>).



Unutar tablice bit će prikazane adrese koje se nalaze u datoteci *Adresar.txt*.

14. U PHP Designeru otvorite datoteku *SpremiAdresu.php* koja se nalazi u mapi "…\D350\vjezbe\vjezba9.2.

15. Na kraj datoteke, prije završne oznake body, dodajte jednu oznaku za novi red (*br*) i linkove za stranice *UnosAdrese.htm* i *PregledAdresa.php,* odijeljene oznakom za razmak *( )*:

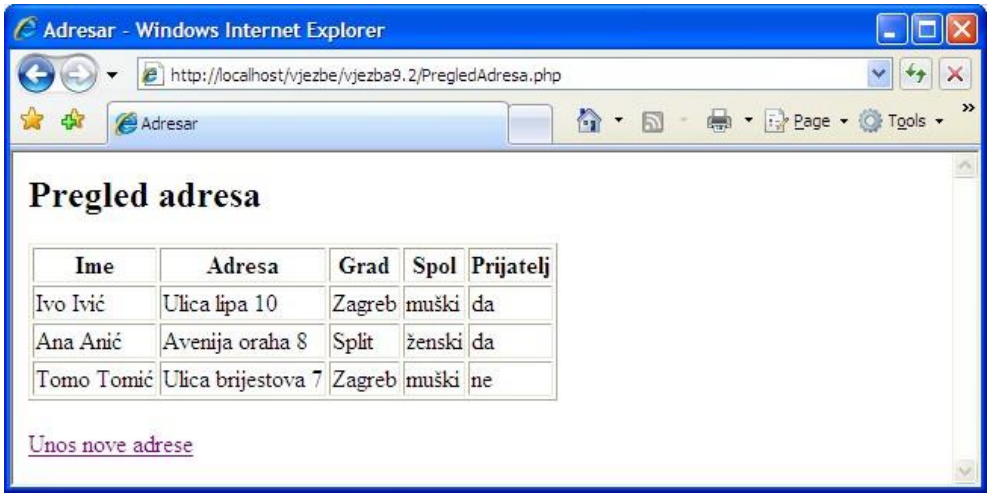
```
  <br/>
  <a href="UnosAdrese.htm">Unos nove adrese</a>
   
  <a href="PregledAdresa.php">Pregled adresa</a>
</body>
```

16. Spremite datoteku.

17. U web pregledniku, na stranici *PregledAdresa.php* kliknite na link "Unos nove adrese".

18. Unesite podatke u obrazac i kliknite "Spremi".

19. Kliknite na link za pregled adresa.



Novounesena adresa bit će prikazana u pregledu adresa.

Vježba 9.3 – Slanje datoteke na poslužitelj

Ova vježba se nastavlja na prethodnu vježbu.

1. Pokrenite program *PHP Designer.* U mapi "…\D350\vjezbe\vjezba9.3" otvorite datoteku *UnosAdrese.htm*. Unutar oznake *form* dodajte masno otisnuti kôd:

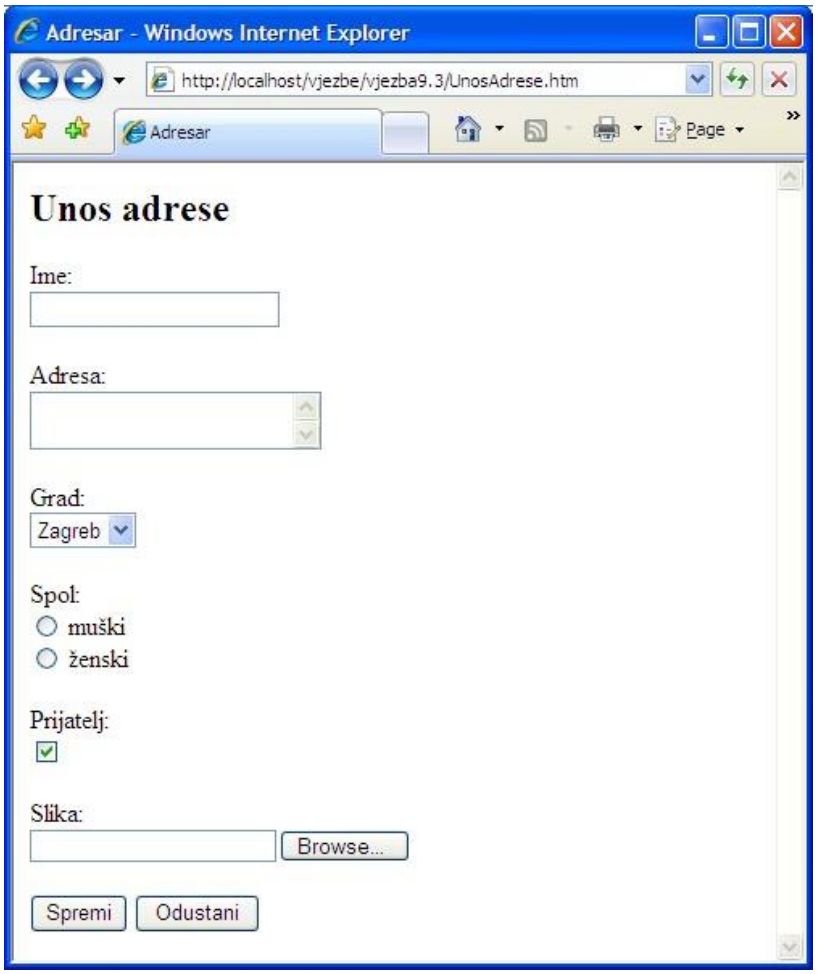
```
<form method="POST" action="SpremiAdresu.php"
 enctype="multipart/form-data" >
```

Atribut *enctype* postavljen na vrijednost "multipart/form-data" omogućuje ispravno slanje datoteka na poslužitelj.

2. Između kućice za označavanje kvačicom i dugmeta "Spremi", dodajte novi element u obrazac – polje za odabir datoteke, zajedno s opisnim tekstom i nekoliko oznaka *br* (radi estetskih razloga).

```
Prijatelj:<br/>
<input type="checkbox" checked="checked" name="prijatelj"/>
<br/><br/>
Slika:<br/>
<input type="file" name="slika"/>
<br/><br/>
<input type="submit" value="Spremi"/> <input type="reset"
value="Odustani"/>
```

3. Spremite datoteku.
4. Pokrenite servis *EasyPHP* (ako već nije pokrenut).
5. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba9.3/UnosAdrese.htm* .



Obrazac za unos adrese sada će izgledati kao na slici.

6. U programu *PHP Designer* otvorite datoteku *SpremiAdresu.php* koja se nalazi u mapi "…\D350\vjezbe\vjezba9.3". Unutar datoteke, prije završne PHP oznake ("?>"), dodajte ovaj kôd:

```
if ($_FILES["slika"]["name"] &&
    $_FILES["slika"]["size"] != 0)
{
   // spremi sliku
}

?>
```

Provjera je li datoteka poslana na poslužitelj može se obaviti ispitivanjem postoji li element *$_FILES["slika"]["name"]* u asocijativnom polju *$_FILES*. Također, postojanje datoteke može se dodatno potvrditi provjerom je li veličina poslane datoteke (element *$_FILES["slika"]["size"]*) različit od nule. Ukoliko datoteka nije poslana, pokušaj spremanja bi rezultirao pogreškom, pa je zato potrebna provjera.

7. Unutar gornjeg *if* bloka dodajte masno otisnutu naredbu:

```
if ($_FILES["slika"]["name"] &&
    $_FILES["slika"]["size"] != 0)
{
   // spremi sliku
   move_uploaded_file($_FILES["slika"]["tmp_name"],
   "Slike/$ime.jpg");
}
```

Funkcija *move_uploaded_file* premjestit će poslanu datoteku s njenog privremenog mjesta (dostupnog u polju *$_FILES*, preko elementa *$_FILES["slika"]["tmp_name"]*) na novo mjesto u mapi "Slike" (unutar trenutne mape, dakle "…\D350\vjezbe\vjezba9.3\Slike"). Datoteka će biti spremljena pod imenom jednakim imenu osobe koja se upisuje u adresar. Uz to se ime dodaje i ekstenzija .jpg, što znači da će ovaj jednostavan primjer raditi ispravno samo za JPG datoteke.

8. Spremite datoteku.

9. Ispunite obrazac. Za sliku odaberite jednu od slika iz mape "…\D350\vjezbe\materijali\adresar". Kliknite na "Spremi".

10. Otvorite mapu "…\D350\vjezbe\vjezba9.3\Slike". Unutar mape nalazit će se slika s nazivom jednakim imenu osobe koje je bilo uneseno u obrazac.

11. U programu *PHP Designer* otvorite datoteku *PregledAdresa.php* koja se nalazi u mapi "…\D350\vjezbe\vjezba9.3". Unutar zaglavlja tablice koja prikazuje unesene adrese dodajte još jednu ćeliju:

```
<table border="1">
   <tr>
      <th>Ime</th>
      <th>Adresa</th>
      <th>Grad</th>
      <th>Spol</th>
      <th>Prijatelj</th>
      <th>Slika</th>
   </tr>
```

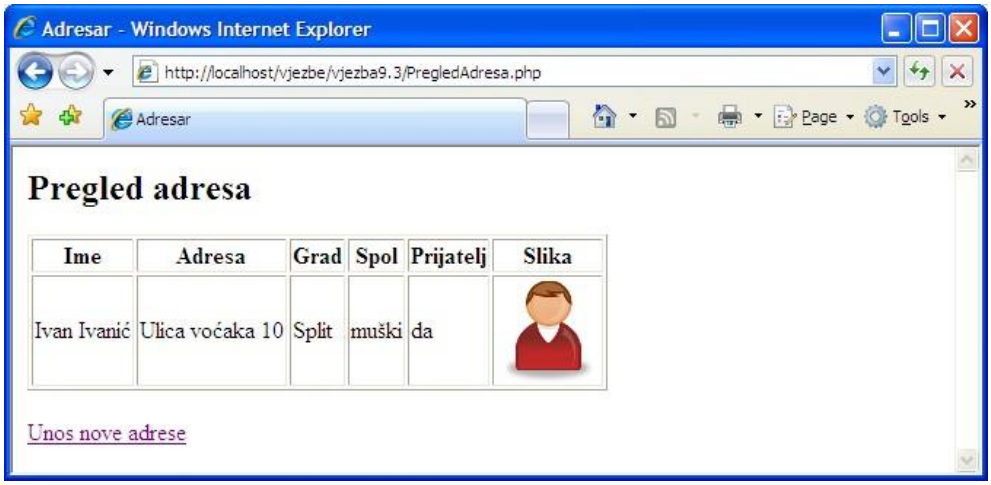
12. Među naredbe koje ispisuju pojedini redak tablice dodajte još jednu:

```
echo "<td>$spol</td>";
echo "<td>$prijatelj</td>";
echo "<td><img src='Slike/$ime.jpg'></td>";
echo "</tr>";
```

Ova naredba ispisuje, unutar ćelije, oznaku img pomoću koje će se prikazati slika koja ima odgovarajuće ime (jednako imenu osobe).

13. Spremite datoteku.

14. U web pregledniku kliknite na link "Pregled adresa."



Unutar tablice bit će, zajedno s ostalim podacima, prikazana i slika koja je poslana na poslužitelj putem obrasca.

U ovom je poglavlju obrađeno:

- otvaranje datoteke pomoću funkcije *fopen*
- zatvaranje datoteke pomoću funkcije *fclose*
- čitanje iz datoteke pomoću funkcije *fread*
- čitanje jednog retka iz datoteke pomoću funkcije *fgets*
- čitanje jednog znaka iz datoteke pomoću funkcije *fgetc*
- ispitivanje je li pročitan znak za kraj datoteke pomoću funkcije *feof*
- pisanje u datoteku pomoću funkcije *fwrite*
- slanje datoteke na server pomoću elementa obrasca *input* s atributom *type="file"*
- spremanje poslane datoteke na serveru pomoću funkcije *move_uploaded_file*

10. Slanje poruke elektroničke pošte

Slanje poruka elektroničke pošte koristi se u web aplikacijama koje služe za pregledavanje elektroničke pošte, ali i u drugim aplikacijama koje se koriste tom mogućnosti za automatsko obavještavanje korisnika.

Pomoću PHP-a moguće je na jednostavan način poslati poruku elektroničke pošte. Poruka može biti poslana automatski (bez interakcije s korisnikom) ili može biti poslana kad korisnik upiše tekst poruke u obrazac i klikne na dugme za slanje podataka.

10.1. Slanje poruke elektroničke pošte

Za slanje poruke elektroničke pošte u jeziku PHP se koristi funkcija *mail*. Argumenti koje ona prima su:

- adresa primatelja (ukoliko je primatelja više, adrese moraju biti odvojene zarezom); adresa može biti u jednom od ova dva oblika:
  - ime.prezime@domena.com
  - Ime Prezime <ime.prezime@domena.com>
- naslov poruke – u naslovu poruke ne smiju se nalaziti znakovi za novi red
- tekst poruke – kao znak za novi red, u tekstu poruke se treba koristiti znak "\n", a linija može sadržavati maksimalno 70 znakova
- dodatna zaglavlja (opcionalno) – pošiljatelj poruke (*From*), dodatni primatelji (*Cc*, *Bcc*) i druga; zaglavlja moraju biti odvojena kombinacijom znakova "\r\n"
- dodatni parametri (opcionalno) – služe za proslijeđivanje parametara programu na poslužitelju koji šalje poruku

Funkcija *mail* vraća vrijednost *TRUE* ako je poruka uspješno poslana, a *FALSE* ako slanje nije uspjelo.

Adresa pošiljatelja koja se navodi prilikom slanja poruke nigdje se neće provjeravati. Ta adresa ne mora biti postojeća, no svakako se ne preporuča upotrebljavati nečiju tuđu adresu, jer će tako poslane poruke izgledati kao da su došle od te osobe.

Primjer slanja poruke izgledao bi ovako:

```
<?php
  $primatelj = 'tecaj00@baltazar.srce.hr';
  $naslov = 'Naslov poruke';
  $tekst = 'Pozdrav svima!';
  $zaglavlja = 'From: tecaj01@baltazar.srce.hr' .
        "\r\n" . 'Cc: tecaj02@baltazar.srce.hr'

  mail($primatelj, $naslov, $tekst, $zaglavlja );
?>
```

10.2. Podešavanje postavki za slanje poruke

Prije slanja poruke moguće je podesiti postavke za slanje poruke elektroničke pošte. Pregled postavki dan je u sljedećoj tablici:

Naziv postavke	Objašnjenje
SMTP	naziv ili IP adresa poslužitelja elektroničke pošte (ova postavka je dostupna samo pod operacijskim sustavom Windows)
smtp_port	port preko kojeg se šalje elektronička pošta (najčešće 25; ova postavka je dostupna samo pod operacijskim sustavom Windows)
sendmail_from	adresa s koje se šalje poruka
sendmail_path	putanja programa za slanje poruka elektroničke pošte na poslužitelju

Ove postavke nalaze se u datoteci *php.ini*. Za podešavanje ovih i drugih konfiguracijskih postavki iz kôda može se koristiti funkcija *ini_set*. Argumenti koje ona prima su:

- naziv postavke
- vrijednost postavke

Ako je postavka uspješno promijenjena, funkcija vraća staru vrijednost postavke, a u suprotnom vrijednost *FALSE*.

Podešavanje poslužitelja elektroničke pošte i adrese pošiljatelja, te potom slanje poruke, izgledat će ovako:

Ukoliko na računalu koje izvršava PHP (npr. lokalno računalo) nije instaliran poslužitelj elektroničke pošte, moguće je koristiti poslužitelj ISP-a. Naravno, to se može koristiti samo u testne svrhe. (Ne u stvarnim primjenama.)

```
<?php
  ini_set("SMTP", "baltazar.srce.hr");
  ini_set("sendmail_from",
          "tecaj01@baltazar.srce.hr");

  mail($primatelj, $naslov, $tekst);
?>
```

U prvoj je naredbi, kao poslužitelj koji će poslati poruku, zadan poslužitelj Srca.

U drugoj naredbi je za predodređenu adresu pošiljatelja postavljena adresa "posiljatelj@srce.hr". Treća naredba pokreće slanje poruke, a kako je pošiljatelj već postavljen, nije ga potrebno navesti u dodatnom (*From*) zaglavlju, pa se u pozivu funkcije izostavlja opcionalni argument u kojem se navode dodatna zaglavlja.

Vježba 10.1 – Slanje podataka iz obrasca putem poruke elektroničke pošte

Ova vježba se nastavlja na vježbu 9.3. iz prethodnog poglavlja.

U adresar iz prethodnih vježbi bit će dodano tekstualno polje za spremanje adrese elektroničke pošte, a ta adresa će se, u obliku linka, prikazivati prilikom pregleda liste unosa. Klikom na link otvorit će se novi obrazac pomoću kojeg će se moći poslati poruka na navedenu adresu.

1. Pokrenite program *PHP Designer.* U mapi "…\D350\vjezbe\vjezba10.1" otvorite datoteku *UnosAdrese.htm*. U datoteku dodajte masno otisnuti dio kôda:

```
</select>
<br/><br/>
E-mail adresa:<br/>
<input type="text" name="email">
<br/><br/>
Spol:<br/>
```

   Spremite datoteku.

2. U programu *PHP Designer* otvorite datoteku *SpremiAdresu.php*. U datoteku dodajte masno otisnutu naredbu:

```
$grad = $_POST['grad'];
$email= $_POST['email'];
```

3. Gornja naredba će pročitati vrijednost s nazivom *email* poslanu pomoću *POST* metode i spremiti je u varijablu *$email*.

4. Među naredbe koje ispisuju pročitane vrijednosti, dodajte i sljedeću masno otisnutu naredbu:

```
    echo "Grad: $grad<br/>";
    echo "E-mail adresa: $email<br/>";
    echo "Spol: $spol<br/>";
```

5. U datoteci promijenite naredbu *fwrite* kako je naznačeno ovdje:

```
fwrite($datoteka,
 "\n$ime\n$adresa\n$grad\n$email\n$spol\n$prijatelj");
```

   Uz ostale vrijednosti, u datoteku *Adresar.txt* bit će zapisana i adresa elektroničke pošte. Spremite izmijenjenu datoteku.

6. U programu *PHP Designer* otvorite datoteku *PregledAdresa.php*. U datoteci, unutar zaglavlja tablice, dodajte masno otisnutu oznaku *th*:

```
<th>Ime</th>
<th>Adresa</th>
<th>Grad</th>
<th>E-mail</th>
<th>Spol</th>
<th>Prijatelj</th>
<th>Slika</th>
```

7. Prilikom čitanja iz datoteke, potrebno je pročitati i adresu elektroničke pošte, pa dodajte sljedeću (masno otisnutu) naredbu u datoteku:

```
$grad = fgets($datoteka);
$email = fgets($datoteka);
$spol = fgets($datoteka);
```

8. Među naredbe koje ispisuju pojedini redak tablice, dodajte naredbu koja će ispisati adresu elektroničke pošte (i to kao link koji će voditi na obrazac za slanje poruke):

```
echo "<td>$grad</td>";
echo "<td><a href='UnosPoruke.php?email=$email'>
       $email</a></td>";
echo "<td>$spol</td>";
```

   Link će voditi na stranicu *UnosPoruke.php*, a kao parametar u URL-u (dakle, metodom *GET*) proslijedit će se adresa na koju treba poslati poruku, sadržana u varijabli $*email*.
   (Napomena: naredbu treba napisati u jednom retku.)

9. U programu *PHP designer* stvorite novu datoteku *UnosPoruke.php*. Ova će datoteka sadržavati obrazac za upisivanje poruke koja će se poslati. Konačna stranica treba izgledati ovako:

10. Unutar stvorene datoteke dodajte oznaku *head* s oznakama *title i meta*, i zatim oznake *body* i *h2*:

```
<html>
<head>
  <title>Adresar</title>
  <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
</head>
<body>
  <h2>Unos poruke</h2>

</body>
</html>
```

11. Unutar oznake *body* dodajte oznaku za obrazac (*form*):

```
<body>
  <h2>Unos poruke</h2>
  <form method="POST" action="PosaljiPoruku.php">

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

Obrazac će se koristiti metodom *POST* za slanje podataka, a skripta koja će obrađivati poslane podatke bit će *PosaljiPoruku.php*.

Unutar obrasca dodajte dva elementa za unos:

```
<form method="POST" action="PosaljiPoruku.php">
  Naslov:<br/>
  <input type="text" name="naslov" />
  <br/><br/>
  Tekst poruke:<br/>
  <textarea name="tekst" rows="4"></textarea>
  <br/><br/>
</form>
```

Obrazac će imati dva elementa za unos – polje za unos teksta u koje se upisuje naslov poruke, i višelinijsko polje za unos teksta u koje se upisuje tekst poruke.

12. Unutar obrasca dodajte i skriveno polje (oznaka *input* s atributom *type* postavljenim na *hidden*):

```
  <textarea name="tekst" rows="4"></textarea>
  <br/><br/>
  <input type="hidden" name="email"
         value="<?php echo $_GET['email'] ?>">
</form>
```

Uloga ovog polja je da se adresa elektroničke pošte proslijedi skripti *PosaljiPoruku.php* zajedno s podacima unesenim u obrazac.

Vrijednost atributa *value* skrivenog polja mora se postaviti dinamički, jer se unaprijed ne zna koja će biti odabrana adresa. To se postiže tako da se na mjesto gdje treba biti ispisana vrijednost atributa stave oznake za PHP kôd, a unutar njih se pomoću naredbe *echo* ispisuje željena vrijednost.

Odabrana adresa bit će dostupna kao parametar u URL-u, pa se može pročitati iz polja *$_GET*.

13. Nakon skrivenog polja, unutar obrasca dodajte još i dugme za slanje podataka i dugme za brisanje podataka iz obrasca:

```
  <input type="hidden" name="email"
         value="<?php echo $_GET['email']?>">
  <input type="submit" value="Pošalji poruku"/>
  <input type="reset" value="Odustani"/>
</form>
```

14. Na kraju dodajte još i linkove na stranice za unos nove adrese i pregled unesenih adresa.

```
  </form>
  <br/>
  <a href="UnosAdrese.htm">Unos nove adrese</a>
   
  <a href="PregledAdresa.php">Pregled adresa</a>
</body>
</html>
```

15. Spremite datoteku.

16. U programu *PHP designer* stvorite novu datoteku naziva *PosaljiPoruku.php*. Ova će skripta primati podatke s obrasca i izvršavati slanje poruke elektroničke pošte.

    U datoteku upišite sljedeći HTML kôd:

```
<html>
<head>
  <title>Adresar</title>
  <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
</head>
<body>
  <h2>Slanje poruke</h2>

</body>
</html>
```

    Radi se o oznaci *head* i oznaci *h2* s naslovom koji će se prikazati.

17. U datoteku upišite sljedeće (masno otisnute) PHP naredbe, unutar oznaka za PHP:

```
  <h2>Slanje poruke</h2>
  <?php
    $email= $_POST['email'];
    $naslov = $_POST['naslov'];
    $tekst = $_POST['tekst'];

    echo "Primatelj: $email<br/>";
    echo "Naslov: $naslov<br/>";
    echo "Tekst poruke: $tekst<br/><br/>";
  ?>
</body>
</html>
```

Prve tri naredbe čitaju podatke proslijeđene *POST* metodom, dostupne unutar polja *$_POST*, i spremaju ih u varijable *$email*, *$naslov* i *$tekst*. Sljedeće tri varijable ispisuju pročitane vrijednosti (zajedno s opisnim tekstom).

18. Nakon ovih naredbi upišite u datoteku naredbe za podešavanje postavki za slanje elektroničke pošte:

```
    echo "Tekst poruke: $tekst<br/><br/>";

    ini_set("SMTP", "baltazar.srce.hr");
    ini_set("sendmail_from", "ime.prezime@domena.hr");
  ?>
```

Prva naredba zadaje naziv poslužitelja koji će slati poruke. U tu svrhu se u ovoj vježbi koristi poslužitelj Srca. Ako on nije dostupan (ako radite ovu vježbu kod kuće), možete koristiti poslužitelj ISP-a preko kojeg ste spojeni na Internet.

19. Nakon ovih naredbi dodajte naredbe za slanje poruke i za ispisivanje odgovarajućeg teksta ako je slanje uspjelo:

```
  ini_set("sendmail_from", "ime.prezime@srce.hr");

  if (mail($email, $naslov, $tekst))
  {
     echo "Poruka je uspješno poslana.";
  }
  else
  {
     echo "Poruka nije uspješno poslana.";
  }
?>
```

Adresu pošiljatelja nećemo navoditi jer je već prije postavljena pomoću funkcije *ini_set*. Ako slanje poruke uspije, funkcija će vratiti logičku vrijednost TRUE i ispisat će se tekst "Poruka je uspješno poslana". U suprotnom, ispisat će se drugi tekst.

20. Na kraju dodajte još i linkove na stranice za unos nove adrese i pregled unesenih adresa (radi omogućavanja povratka).

```
  ?>
  <br/><br/>
  <a href="UnosAdrese.htm">Unos nove adrese</a>
   
  <a href="PregledAdresa.php">Pregled adresa</a>
</body>
</html>
```

21. Spremite datoteku.

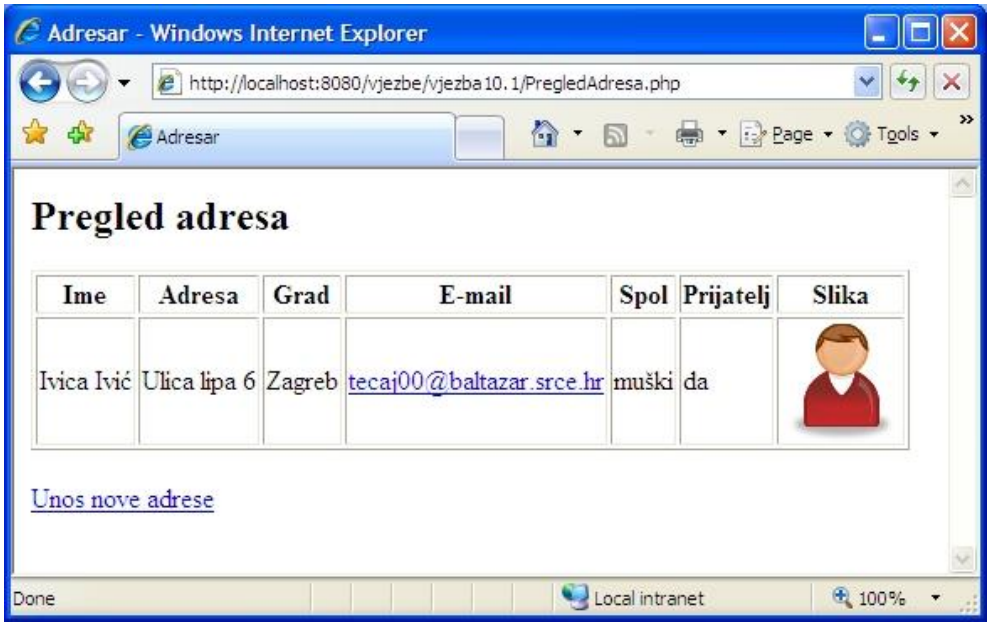
22. Pokrenite servis *EasyPHP* (ako već nije pokrenut).

23. U web preglednik upišite adresu:
    http://localhost/vjezbe/vjezba10.1/UnosAdrese.htm .

24. Na obrascu za unos u adresar unesite ime, prezime i ostale podatke. Kao adresu elektroničke pošte upišite adresu *tecaj00@baltazar.srce.hr*. Ako želite, odaberite i sliku. Kliknite na "Spremi".

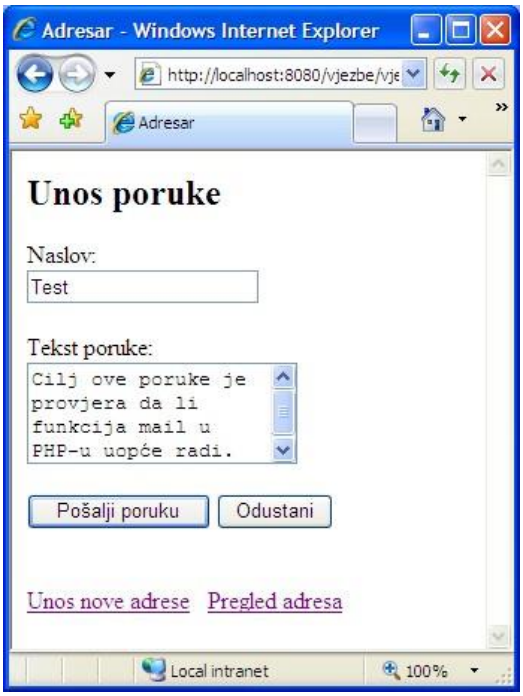
25. Na stranici s potvrdom da je adresa spremljena, kliknite na link "Pregled adresa".

26. Na stranici s pregledom adresa, primjećujete da je ispisana adresa elektroničke pošte zapravo link na drugu stranicu (*UnosPoruke.php*) i da je sama adresa parametar u tom linku.



Kliknite na taj link.

27. Upišite naslov i tekst poruke. Kliknite na dugme "Pošalji poruku".



28. Ako je slanje poruke uspjelo, prikazat će se stranica s obavijesti da je poruka uspješno poslana.

U ovom je poglavlju obrađeno:

- slanje poruke elektoničke pošte pomoću funkcije *mail*
- podešavanje postavki za slanje elektroničke pošte pomoću funkcije *ini_set*
- slanje podataka unesenih u obrazac putem elektroničke pošte

11. Rad s MySQL bazama podataka

S MySQL bazama podataka može se raditi izravnim upisivanjem naredbi u naredbenom retku, ili korištenjem nekog klijentskog programa. Jedan od najčešćih načina je uporaba aplikacije *phpMyAdmin* koja obično dolazi uz instalaciju sustava MySQL.

11.1. Aplikacija *phpMyAdmin*

PhpMyAdmin je aplikacija napisana u jeziku PHP koja služi za upravljanje MySQL bazama podataka. Pomoću ove aplikacije moguće je stvoriti novu bazu podataka, novu tablicu, pregledavati i mijenjati podatke i tipove podataka u tablicama, te brojne druge akcije potrebne za rad s bazom podataka.

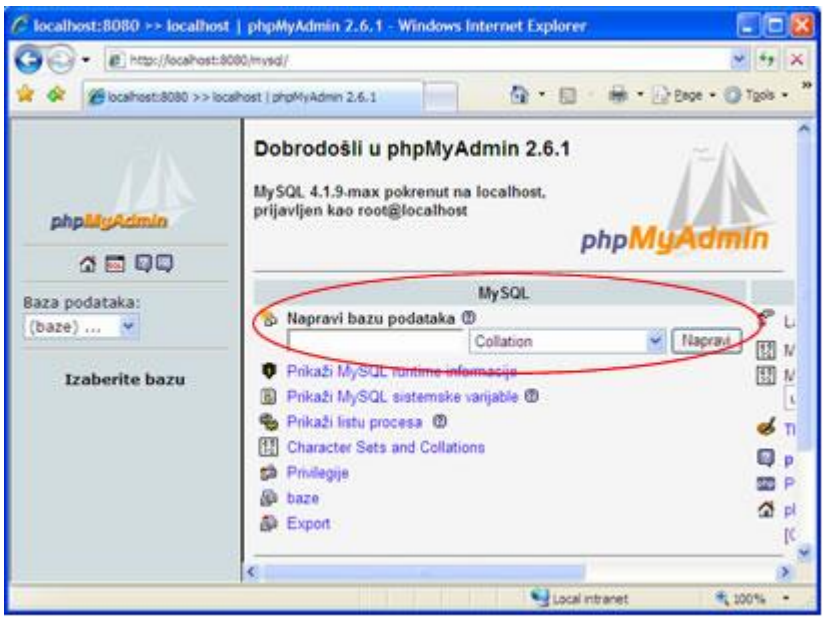
Aplikacija se pokreće u web pregledniku upisivanjem adrese http://localhost/mysql/.

11.2. Stvaranje baze podataka

Za svaki projekt potrebna je odvojena baza podataka. Nova baza podataka može se napraviti pomoću ove SQL naredbe:

```
CREATE DATABASE imeBaze
```

Preko aplikacije *phpMyAdmin* također se može stvoriti nova baza podataka. Na početnoj stranici aplikacije nalazi se dio s naslovom "MySQL" u kojem se, zajedno s linkovima na brojne postavke, nalazi obrazac za stvaranje nove baze podataka, Potrebno je upisati naziv nove baze i kliknuti na dugme "Napravi".



Prilikom stvaranja nove baze podataka je potrebno odabrati *collation,* odnosno način abecednog sortiranja znakovnih nizova (koji se razlikuje kod raznih jezika). Odabirom *collationa* u aplikaciji *phpMyAdmin* odabire se i

odgovarajuća kôdna stranica, što određuje na koji način će se znakovi specifični za određeni jezik (npr. dijakritički znakovi) pretvoriti u binarne podatke.

Collation i kôdna stranica mogu se podesiti na razini poslužitelja, baze, tablice ili stupca u tablici.

11.3. Stvaranje tablica

Baza podataka se sastoji od međusobno povezanih tablica. Za svaku vrstu entiteta koji će se pohranjivati u bazu podataka, stvara se odgovarajuća tablica koja će sadržavati takve entitete.

Tablice su definirane poljima koje sadrže, i koja opisuju entitet. Svaki zapis koji se sprema u bazu podataka, zapisuje se kao redak u odgovarajuću tablicu.

Za jednostavni adresar koji je razrađivan u vježbama uz prethodna poglavlja, može se definirati entitet *kontakt*. Kontakti se mogu zapisivati u ovakvu tablicu:

Id	Ime	Adresa	Grad	Email	Spol	Prijatelj
1	Ivica Ivić	Ulica lipa 10	Zagreb	iivic@srce.hr	M	Da
2	Tomica Tomić	Ul. jorgovana 5	Split	ttomic@srce.hr	M	Ne
3	Marica Marić	Avenija vrbi b.b.	Zagreb	mmaric@srce.hr	Ž	Da

Polje *Id* u gornjem primjeru služi kao jedinstveni identifikator retka (primarni ključ). Poželjno je da svaka tablica ima primarni ključ – polje čije su vrijednosti jedinstvene za svaki pojedini redak i na temelju kojeg se redak može jedinstveno dohvatiti. Primarni ključ može biti definiran i preko više polja, a pritom kombinacija vrijednosti tih polja mora biti jedinstvena za svaki redak.

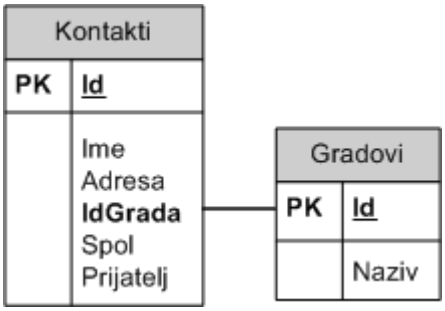
U dizajnu baza podataka teži se tome da se podaci ne ponavljaju na više mjesta. Jedan podatak bi se trebao nalaziti samo na jednom mjestu (radi uštede diskovnog prostora, ali i radi toga što se, u slučaju promjene, promjena podatka treba izvršiti samo na jednom mjestu). U gornjem primjeru nazivi gradova bit će isti kod velikog broja kontakata. Stoga bi se grad mogao izdvojiti kao zasebni entitet u vlastitu tablicu:

Id	Naziv
1	Zagreb
2	Split

Tablica s kontaktima sada bi izgledala ovako:

Id	Ime	Adresa	IdGrada	Email	Spol	Prijatelj
1	Ivica Ivić	Ulica lipa 10	**1**	iivic@srce.hr	M	Da
2	Tomica Tomić	Ul. jorgovana 5	**2**	ttomic@srce.hr	M	Ne
3	Marica Marić	Avenija vrbi b.b.	**1**	mmaric@srce.hr	Ž	Da

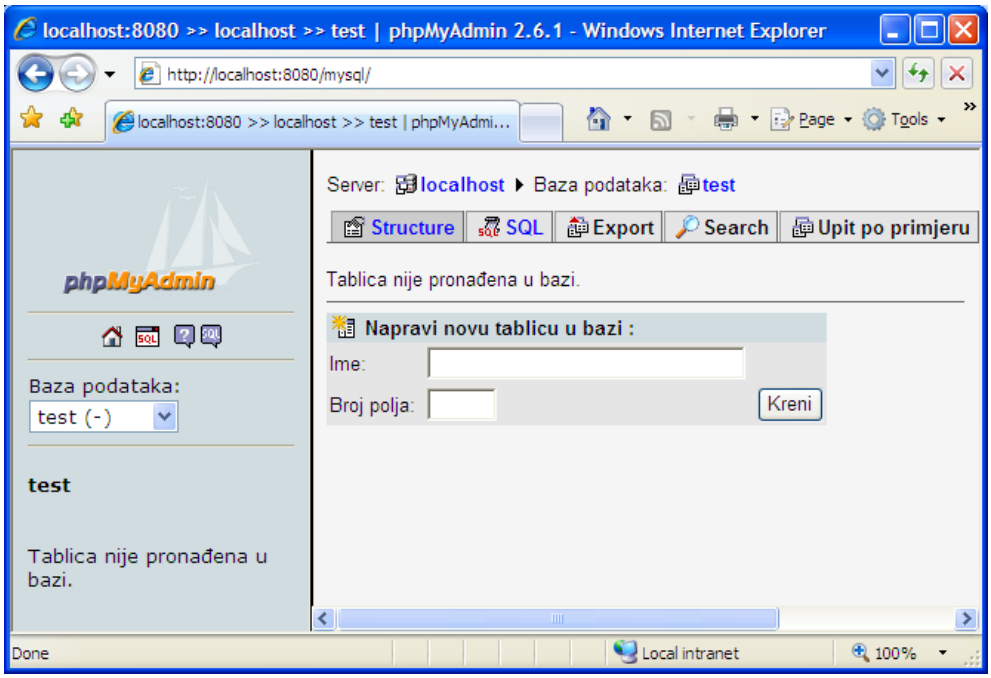
Tablica s kontaktima i tablica s gradovima sad su povezane preko identifikatora grada, a nazivi gradova se nalaze u zasebnoj tablici. Dijagram ove baze podataka izgledat će ovako:



Nova tablica stvara se pomoću naredbe CREATE TABLE, a pritom je potrebno navesti sva polja tablice i njihove tipove podataka.

```
CREATE TABLE imeTablice (
   nazivPolja1 TIP_PODATKA,
   nazivPolja2 TIP_PODATKA
);
```

Ako se nova tablica stvara preko aplikacije *phpMyAdmin*, prvi korak je zadavanje imena tablice. U lijevom okviru aplikacije potrebno je odabrati bazu, te potom u desnom okviru upisati naziv tablice i broj polja koja će tablica imati, i pritisnuti na dugme "Kreni".


localhost:8080 >> localhost >> test | phpMyAdmin 2.6.1 - Windows Internet Explorer
http://localhost:8080/mysql/
localhost:8080 >> localhost >> test | phpMyAdmi...
Page
Tools
phpMyAdmin
Baza podataka:
test (-)
test
Tablica nije pronađena u bazi.
Server: localhost ▸ Baza podataka: test
Structure
SQL
Export
Search
Upit po primjeru
Tablica nije pronađena u bazi.
Napravi novu tablicu u bazi :
Ime:
Broj polja:
Kreni
Done
Local intranet
100%

11.4. Tipovi podataka

Sljedeći korak u stvaranju tablice je definiranje tipova podataka kojima će pojedina polja koristiti. Pregled tipova podataka koje koristi MySQL dan je u sljedećoj tablici:

Tip podatka	Opis
INT	cijeli broj uobičajene veličine
TINYINT	vrlo malen cijeli broj (raspon -128 do 127)
SMALLINT	malen cijeli broj
MEDIUMINT	srednje velik cijeli broj
BIGINT	veliki cijeli broj
FLOAT	decimalni broj s pomičnim zarezom
DOUBLE	decimalni broj s pomičnim zarezom (dvostruka preciznost)
DECIMAL	decimalni broj
DATE	datum (format YYYY-MM-DD)
DATETIME	datum i vrijeme (format YYYY-MM-DD HH:MM:SS)
TIMESTAMP	vremenska oznaka (format YYYY-MM-DD HH:MM:SS)
TIME	vrijeme (format HH:MM:SS)
YEAR	godina (format YYYY ili YY)
CHAR	niz znakova fiksne duljine (maksimalno 255)
VARCHAR	niz znakova varijabilne duljine (također maksimalno 255)
TEXT	tekstualni podaci
TINYTEXT	tekstualni podaci manje veličine
MEDIUMTEXT	tekstualni podaci srednje veličine
LONGTEXT	veliki tekstualni podaci
BINARY	niz bajtova fiksne duljine (do 255)
VARBINARY	niz bajtova varijabline duljine (do 255)
BLOB	binarni podaci (Binary Large OBject)
TINYBLOB	binarni podaci manje veličine (do 255 bajtova)
MEDIUMBLOB	binarni podaci srednje veličine
LONGBLOB	veliki binarni podaci (do 4 GB)
ENUM	enumeracija – vrijednost može biti jedna od predefiniranih vrijednosti
SET	skup – podatak može poprimiti nijednu, jednu ili više predefiniranih vrijednosti

Najčešće korišteni tipovi podataka su INT za cijele brojeve, DOUBLE ili DECIMAL za decimalne brojeve, DATETIME za vrijeme, VARCHAR ili TEXT za znakovne nizove, te BLOB za binarne podatke.

Prilikom definicije tablice, za polja se, uz tip podatka, navodi i veličina polja. Ako se ne navede, koristi se predefinirana veličina za taj tip.

Svako polje, ako se ne definira suprotno, može poprimiti vrijednost NULL, što je posebna konstanta koja označava da polje ne sadrži podatak. Za polja čije vrijednosti trebaju biti obavezno postavljene, potrebno je prilikom definicije navesti *NOT NULL*.

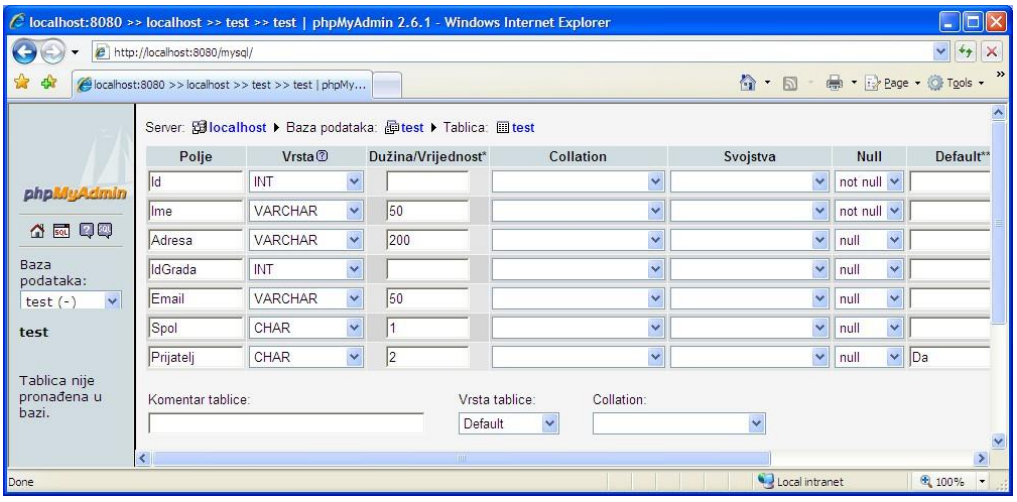
Također je moguće zadati predefiniranu vrijednost polja korištenjem ključne riječi *DEFAULT*.

Primarni ključ se postavlja navođenjem ključnih riječi PRIMARY KEY na kraju definicije polja.

Naredba kojom bi se definirala tablica *kontakti* iz prethodnog primjera izgledala bi ovako:

```
  CREATE TABLE kontakti (
     Id INT NOT NULL ,
     Ime VARCHAR (50) NOT NULL ,
     Adresa VARCHAR (200) ,
     IdGrada INT ,
     Email VARCHAR (50),
     Spol CHAR (1),
     Prijatelj CHAR (2) DEFAULT 'Da',
     PRIMARY KEY (Id)
);
```

Definiranje polja u aplikaciji *phpMyAdmin* izgleda ovako (nakon što se upiše ime tablice i broj polja):



Pomoću lista za odabir za svako se polje treba odabrati tip podataka, i ako je potrebno upisati veličinu polja. Također je u drugoj listi za odabir moguće odabrati može li polje poprimati vrijednost NULL il ne, a u polju za unos u supcu "Default" moguće je unijeti predefiniranu vrijednost za polje.

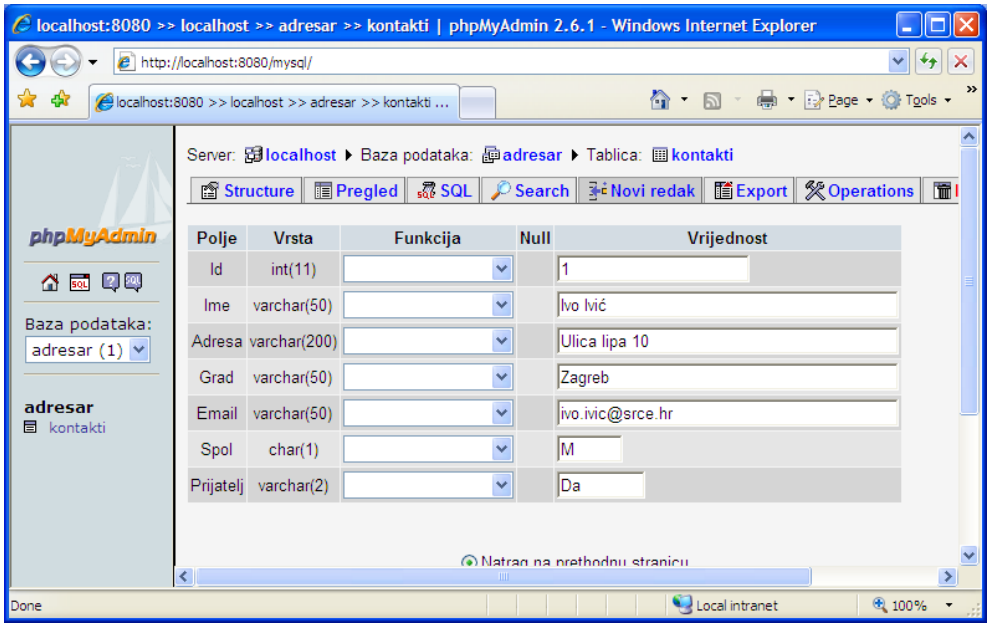
Označavanjem dugmeta za odabir ispod ikone  može se odabrati da određeno polje bude primarni ključ tablice.

Pritiskom na dugme "Spremi" tablica će biti stvorena i bit će vidljiva u popisu tablica u lijevom okviru aplikacije.

11.5. Dodavanje, izmjena i brisanje redaka

Upisivanje novih redaka u tablicu može se obaviti preko aplikacije *phpMyAdmin*. U lijevom okviru aplikacije potrebno je odabrati željenu tablicu i kliknuti na link "Novi redak".

Potrebno je jednostavno upisati željene vrijednosti za svako polje. Aplikacija nudi mogućnost dodavanja dva retka odjednom (u tom slučaju kućica za označavanje "Ignoriraj" ne treba biti označena). Za spremanje upisanih podataka, kliknite na dugme "Kreni".



Za izmjenu ili brisanje retka potrebno je kliknuti na link "Pregled" i u željenom retku odabrati ikonu za izmjenu ( ) ili brisanje ( )

Dodavanje, izmjena i brisanje redaka je moguće i pomoću SQL naredbi, što će biti obrađeno u sljedećem potpoglavlju.

11.6. Izvođenje upita nad bazom podataka

U jeziku SQL postoje četiri vrste upita – *SELECT*, *INSERT*, *UPDATE* i *DELETE*.

SELECT upit služi za dohvaćanje podataka iz tablice. Upit koji bi dohvatio sve podatke iz tablice *kontakti* izgledat će ovako:

```
SELECT * FROM kontakti
```

Zvjezdica u *SELECT* upitu označava da se iz tablice *kontakti* čitaju sva polja. Upit koji dohvaća samo neka polja izgledat će ovako:

```
SELECT Ime, Adresa, Grad, Email FROM kontakti
```

Svaki upit može se ograničiti uvjetom, tako da će biti dohvaćeni samo podaci koji zadovoljavaju navedeni uvjet. Uvjete je moguće kombinirati pomoću operatora AND i OR.

```
SELECT * FROM kontakti
  WHERE IdGrada = 1 AND Prijatelj = 'Da'
```

Gornji upit dohvatit će sve kontakte u adresaru koji su iz Zagreba (identifikator grada 1) i vode se kao prijatelji.

Dodavanje novog retka vrši se pomoću naredbe *INSERT*. Upit kojim se dodaje novi redak u tablicu *kontakti* izgleda ovako:

```
INSERT INTO kontakti
  (Id, Ime, Adresa, Grad, Email, Spol, Prijatelj)
  VALUES ( 2, 'Tom Tomić', 'Aleja jorgovana 5', 2,
  'ttomic@srce.hr', 'M', 'Da' )
```

Izmjena retka vrši se pomoću naredbe *UPDATE*:

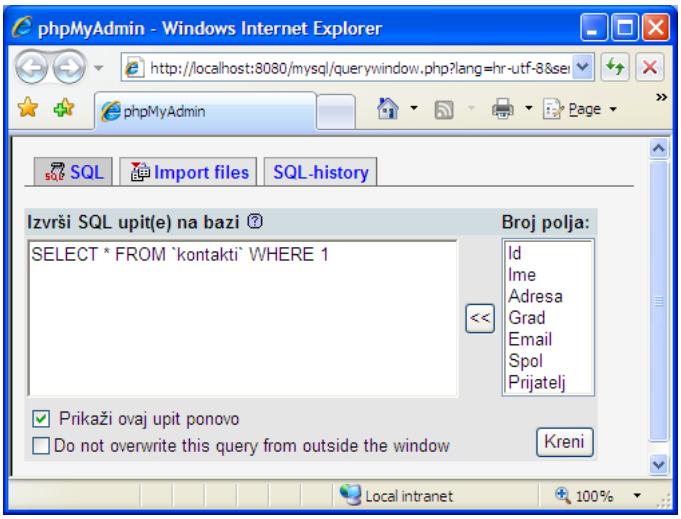
```
UPDATE kontakti SET IdGrada = 1
  WHERE Id = 2
```

Brisanje retka vrši se pomoću naredbe *DELETE*:

```
DELETE from kontakti WHERE Id = 2
```

UPDATE i *DELETE* upiti najčešće sadržavaju uvjet koji određuje na koje retke ta naredba utječe. Ako se ne navede uvjet, svi reci u tablici bit će izmijenjeni, odnosno obrisani.

Da bi se izvršio upit preko aplikacije *phpMyAdmin*, potrebno je najprije odabrati željenu bazu podataka i potom, u lijevom okviru aplikacije, kliknuti na ikonu za izvršavanje SQL upita ().



U novom prozoru koji će se otvoriti potrebno je upisati tekst upita i pritisnuti dugme "Kreni".

11.7. Spremanje podataka iz baze podataka u datoteku

Podatke iz baze podataka moguće je, u svrhu pohrane podataka, spremiti u datoteku. U aplikaciji *phpMyAdmin* potrebno je odabrati link "Export". U lijevom dijelu moguće je odabrati vrstu datoteke u koju će se podaci eksportirati – SQL datoteku (niz naredbi CREATE TABLE i INSERT čijim izvršavanjem se baza podataka može obnoviti), CSV datoteku (vrijednosti odvojene zarezom), XML datoteku ili druge formate.

Klikom na dugme "Kreni", bit će prikazan eksportirani tekst koji je potom potrebno kopirati u datoteku za pohranu.

Vježba 11.1 – Stvaranje baze podataka pomoću aplikacije *phpMyAdmin*

U vježbi se stvara baza podataka koja će se upotrebljavati za spremanje podataka iz aplikacije adresar iz vježbi 8.1 -10.1.

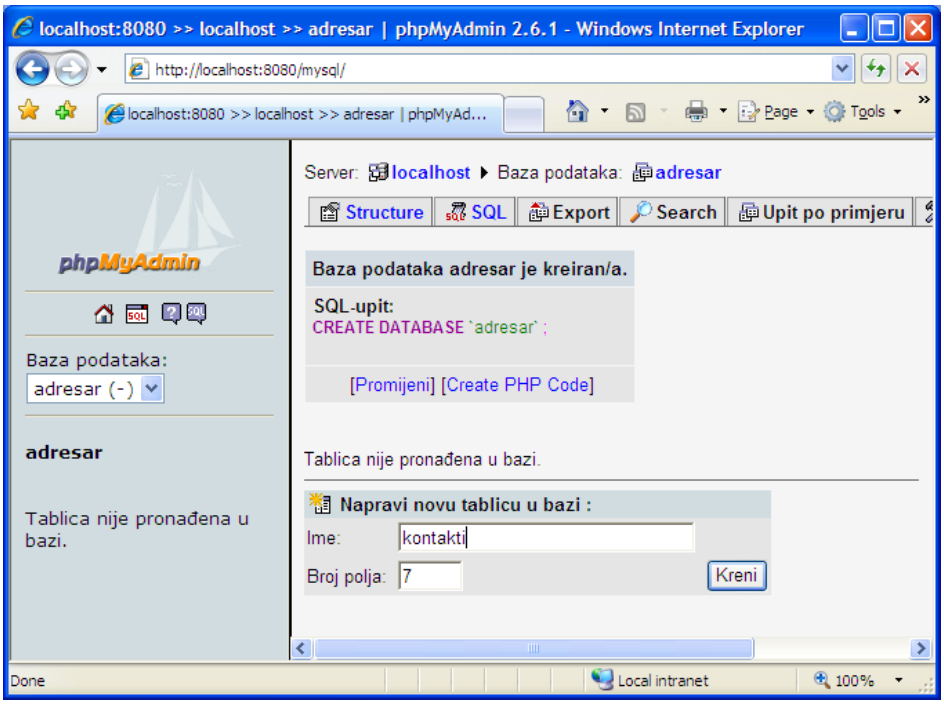
1. Pokrenite servis *EasyPHP*.

2. Pokrenite aplikaciju *phpMyAdmin* tako da u web preglednik upišete adresu http://localhost/mysql/.

3. Stvorite novu bazu podataka. Upišite ime baze u polje za unos teksta (ispod natpisa "Napravi bazu podataka".). Baza će se zvati "adresar". Za *collation* baze odaberite vrijednost *utf8_general_ci*. Kliknite na dugme "Napravi".

Ako je sve u redu, prikazat će se poruka "Baza podataka adresar je kreiran/a." Također je prikazana i poruka "Tablica nije pronađena u bazi.", zato što baza za sada ne sadrži ni jednu tablicu.

4. Stvorite novu tablicu u bazi podataka. Ispod natpisa "Napravi novu tablicu u bazi" upišite ime tablice ("kontakti") i broj polja (7) i kliknite na dugme "Kreni".



5. Stvorite novu tablicu u bazi podataka. Ispod natpisa "Napravi novu tablicu u bazi" upišite ime tablice ("kontakti") i broj polja (7) i kliknite na dugme "Kreni".

6. Tablica će sadržavati sljedeća polja:

Naziv polja	Tip podatka	Duljina
Id	INT	
Ime	VARCHAR	50
Adresa	VARCHAR	200
Grad	VARCHAR	50
Email	VARCHAR	50
Spol	CHAR	2
Prijatelj	CHAR	2

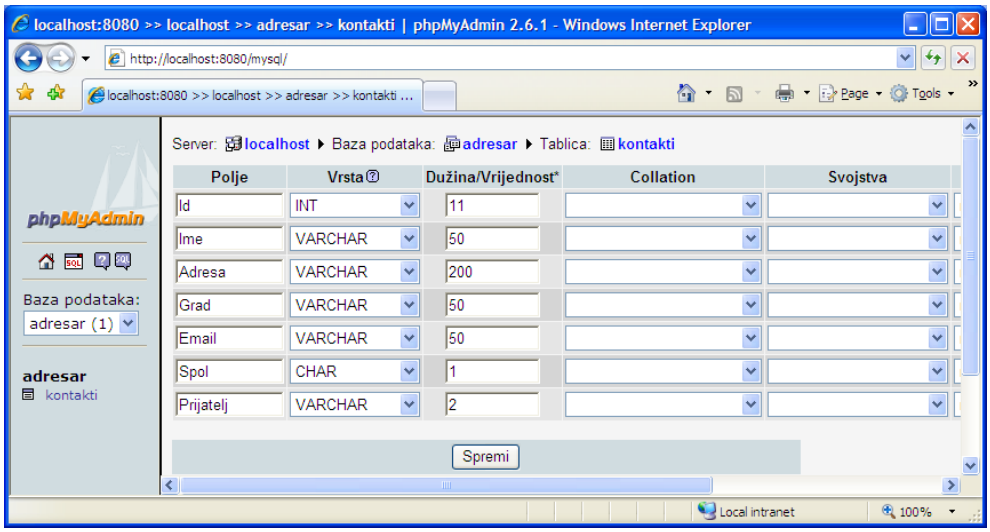
Prvo polje *Id* služit će kao identifikator u bazi podataka (primarni ključ). Vrijednost ovog polja definirat će na jedinstven način redak u tablici, i neće postojati dva retka s istom vrijednosti polja *Id*. *Id* će biti cijeli broj (*INT*). Pri unosu novog retka u bazu podataka, retku će se dodijeliti sljedeći slobodni broj kao *Id*. Prvi redak će tako imati *Id* jednak 1, drugi 2, i tako dalje.

U polja *Ime*, *Adresa*, *Grad* i *Email* zapisivat će se vrijednosti unesene u obrazac za unos adrese. Kako su to tekstualni podaci, koristit ćemo se tipom podatka VARCHAR koji služi za zapisivanje niza znakova. Kao maksimalnu duljinu zadat ćemo 50 znakova, osim za adresu za koju ćemo za maksimalnu duljinu zadati 200 znakova.

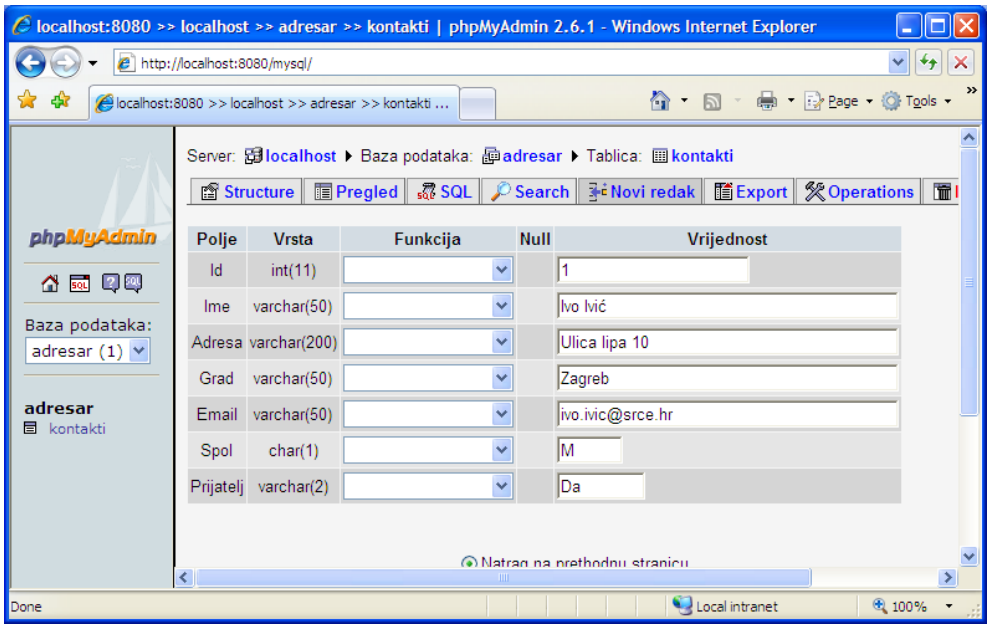
Za polja *Spol* i *Prijatelj* koristit ćemo se tipom podatka CHAR. Podatke za *Spol* zapisivat ćemo kao vrijednosti 'M' za muški i 'Ž' za ženski spol, za što nam treba dva znaka (jer je slovo Ž u kôdnoj stranici utf-8 prikazano pomoću dva znaka). U polje *Prijatelj* upisivat ćemo vrijednosti 'Da' i 'Ne', za što nam također treba dva znaka.

Upišite nazive polja iz gornje tablice i odaberite odgovarajući tip podatka u listi za odabir. Za tekstualna polja upišite i odgovarajuću duljinu. Na polje *Id* postavite primarni ključ klikom na dugme za odabir ispod ikone  . Također za polje *Id* odaberite *auto_increment* pod opcijom "Dodatno". Tako će se vrijednost polja *Id* automatski povećavat za svaki novi redak.

Kliknite na dugme "Spremi".

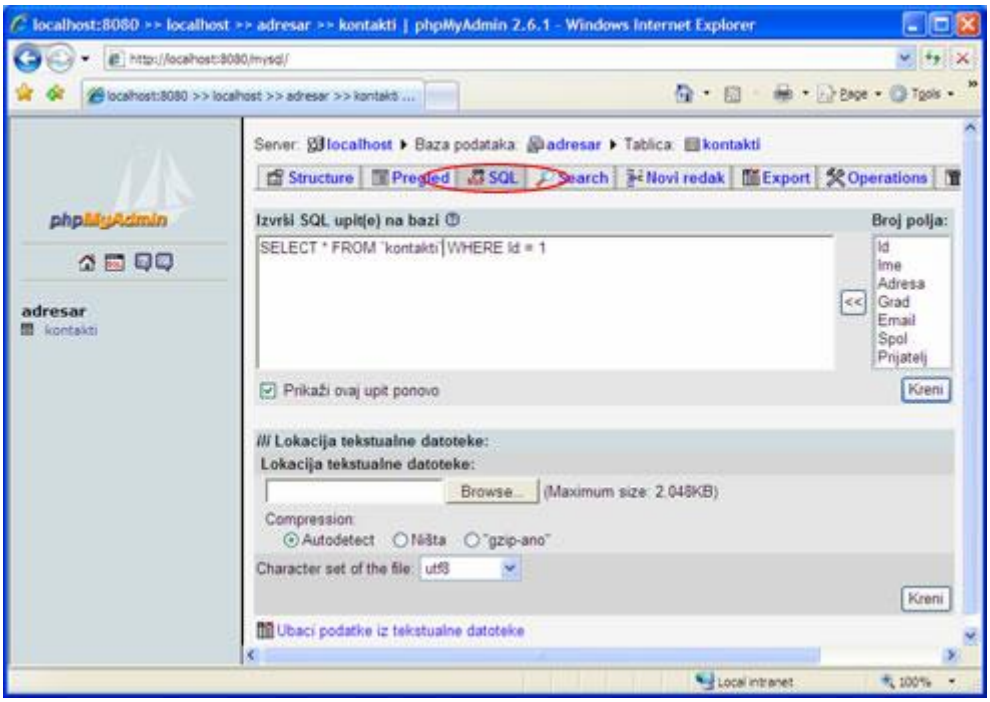


7. Kliknite na link "Novi redak" (u zaglavlju aplikacije). U stupcu "Vrijednost" upišite ime, adresu i ostale podatke. Za polje *Id* nemojte ništa upisivati. Kliknite na dugme "Kreni".



Primjećujete da je uneseni redak automatski dobio 1 kao vrijednost polja *Id*.

8. Kliknite na link "SQL" u zaglavlju aplikacije. Otvara se sučelje za postavljanje SQL upita nad odabranom bazom podataka.



Upišite upit "SELECT * FROM kontakti WHERE Id = 1" i kliknite na dugme "Kreni". Kao rezultat vratit će se prvi upisani redak.

U ovom je poglavlju obrađeno:

- rad s aplikacijom *phpMyAdmin*
- stvaranje baze podataka
- stvaranje tablica
- tipovi podataka u sutavu za upravljanje bazama podataka MySQL
- postavljanje primarnog ključa na tablici
- osnovni upiti u jeziku SQL
- spremanje podataka iz baze podataka u datoteku

12. PHP i MySQL

U ovom poglavlju bit će obrađeno komuniciranje s bazom podataka MySQL iz PHP skripte. Koraci koje je pritom potrebno napraviti su:

- stvaranje veze (konekcije) na MySQL
- odabir baze podataka na koju se spajamo
- izvršavanje SQL upita
- dohvaćanje rezultata (ako smo postavili SELECT upit)
- zatvaranje veze.

12.1. Otvaranje i zatvaranje veze s bazom podataka

Veza ili konekcija je termin koji se koristi u radu s bazama podataka, a označava korisnikov proces koji se spaja na bazu podataka. Sustav za upravljanje bazama podataka MySQL predefinirano dopušta 100 istovremenih veza što znači da 101. korisnik mora čekati da se zatvori neka od prethodno otvorenih veza, odnosno dobiva poruku o preopterećenju. Veza će se zatvoriti automatski nakon prestanka izvršavanja skripte, no dobra je praksa zatvaranje veze nakon prestanka s bazom podataka.

Ovako bi izgledao primjer komunikacije s bazom:

```
<?php
  $veza = mysql_connect("posluzitelj", "korisnik",
          "lozinka");

  mysql_select_db("imebaze", $veza);

/* ovdje dolazi izvrsavanje upita i dohvacanje rezultata */

  mysql_close($veza);
?>
```

Za stvaranje veze koristi se funkcija ***mysql_connect*** kojoj je potrebno zadati ime računala na kojem se baza podataka nalazi, te korisničko ime i lozinku za korisnički račun koji ima pravo pristupa toj bazi podataka na sustavu MySQL. U primjerima i vježbama navedenima u ovom tečaju funkciju *mysql_connect* pozivat ćemo ovako:

```
<?php
  $veza = mysql_connect("localhost", "root", "");
?>
```

Uporaba "prazne" lozinke nikako nije preporučljiva u stvarnim primjenama jer se podatke uvijek želi zaštititi od neovlaštenog pristupa.

Localhost je podrazumijevani naziv lokalnog računala, a korisničko ime "root" i prazna lozinka su predodabrani podaci za pristup bazi podataka.

Funkcija vraća identifikator veze koju je otvorila ili logičku vrijednost *false* u slučaju pogreške.

Za odabir baze podataka koristi se funkcija ***mysql_select_db***, kojoj zadajemo ime baze podataka kao niz znakova i identifikator na otvorenu vezu koji nam je vratila funkcija *mysql_connect*. Ova funkcija također vraća *false* u slučaju pogreške

Zatvaranje veze obavlja funkcija ***mysql_close*** kojoj kao argument zadajemo identifikator na vezu.

Na kraju, može se primijetiti da sve tri funkcije imaju prefiks *mysql*, što znači da pripadaju biblioteci funkcija za komunikaciju s bazom podataka MySQL.

Spomenuto je da funkcije *mysql_connect* i *mysql_select_db* vraćaju false u slučaju pogreške. U praksi je dobro provjeravati je li spajanje na bazu podataka uspjelo, pa ako nije, treba ispisati poruku i prekinuti s izvođenjem daljnjih naredbi. U tu svrhu može se koristiti funkcija ***exit*** koja ispisuje predani tekst i prekida izvođenje skripte.

Kôd koji provjerava je li spajanje na bazu podataka uspjelo izgledao bi ovako:

```
<?php
    $veza = mysql_connect("localhost", "root", "");
    if ($veza)
    {
      if (!mysql_select_db("ime_baze", $veza))
      {
        exit("Baza s tim imenom ne postoji!");
      }
    }
    else
    {
      exit("Ne mogu se spojiti na MySql server!");
    }
?>
```

U gornjem kôdu pokušava se otvoriti veza na MySQL poslužitelj, a potom i odabrati baza podataka na poslužitelju. U slučaju da jedan od koraka nije bio uspješan, ispisuje se poruka i prekida se izvođenje ostatka kôda pomoću funkcije *exit*.

12.2. Ispis podataka iz baze podataka

Za dohvaćanje podataka iz baze podataka koristi se tip upita *SELECT*, a njegovo izvršavanje se obavlja pozivom funkcije ***mysql_query***. Ova funkcija kao argument prima tekst SQL naredbe. U slučaju uspješnog izvršavanja upita, funkcija vraća identifikator rezultata upita, a u slučaju neuspješnog izvršavanja upita logičku vrijednost *FALSE*.

Budući da SELECT upiti najčešće vraćaju više redaka, najbolji način za dohvaćanje podataka je čitanje jednog po jednog retka unutar petlje za što služi funkcija ***mysql_fetch_array*** kojoj se kao argument predaje dobiveni identifikator rezultata.

```
<?php
  $rezultat = mysql_query("SELECT * FROM kontakti");
  while ($redak = mysql_fetch_array($rezultat))
  {
    echo "$redak['Ime'] $redak['Adresa']"
    echo "$redak['Grad'] $redak['Email'] <br/>";
  }
?>
```

Funkcija *mysql_fetch_array* svakim uzastopnim pozivom vraća sljedeći redak iz rezultata, a kad dođe do kraja, vraća *false*, te se prestaje s izvršavanjem petlje. Vrijednostima pojedinih atributa unutar retka može se pristupiti pomoću njihovih naziva (npr. *$redak["Ime"]*) jer je varijabla *$redak* zapravo asocijativno polje.

Da bi gornji upit bio uspješno obavljen, potrebno je prije pozivanje funkcije *mysql_query* obaviti spajanje s bazom podataka, kao što je objašnjeno u prethodnom potpoglavlju, a odmah nakon prestanka čitanja iz baze podataka (nakon izlaska iz petlje) poželjno je zatvoriti vezu s bazom podataka.

Kod *SELECT* upita može se, pomoću funkcije ***mysql_num_rows***, saznati broj redaka koji zadovoljavaju upit. Ovoj funkciji kao atribut treba predati identifikator rezultata upita.

Dobrapraksa je osloboditi memoriju zauzetu dohvaćenim podacima (pogotovo ako dohvaćamo veću količinu podataka) čim smo završili s njihovom obradom. Na taj način, slično kao kod zatvaranja veze, oslobađamo resurse poslužitelja za posluživanje drugih korisnika. To se obavlja pozivom funkcije ***mysql_free_result*** kojoj također predajemo identifikator rezultata. Evo kako izgleda dohvaćanje podataka uporabom ove dvije funkcije:

```
<?php
  $rezultat = mysql_query("SELECT * FROM kontakti");
  while ($redak = mysql_fetch_array($rezultat))
  {
    echo "$redak['Ime'] $redak['Adresa']";
    echo "$redak['Grad'] $redak['Email'] <br/>";
  }
  echo "Dohvaćeno " . mysql_num_rows($rezultat) . " redaka";
  mysql_free_result($rezultat);
?>
```

12.3. Ispis podatka koji zadovoljava uvjet

Za ispis podatka koji zadovoljava neki uvjet bit će dovoljno SQL upitu dodati uvjet *WHERE*.

Ako očekujemo da rezultat upita bude samo jedan redak, umjesto funkcije *mysql_fetch_array* može se koristiti funkcija ***mysql_fetch_row*** kojoj se također, kao argument, predaje dobiveni identifikator rezultata. Ako se koristi ta funkcija, rezultat nije asocijativno polje, već se pojedinim poljima u retku mora pristupiti preko njihovog indeksa.

Dohvaćanje retka koji zadovoljava uvjet da mu je polje *Id* jednako određenoj vrijednosti izgledat će kao u ovom primjeru:

```
<?php
  $rezultat = mysql_query(
    "SELECT * FROM kontakti WHERE Id = 1");
  if ($redak = mysql_fetch_row($rezultat))
  {
    echo "Pronađen je ovaj kontakt: ";
    echo "$redak[1] $redak[2] $redak[3]";
  }
  mysql_free_result($rezultat);
?>
```

I u ovom slučaju je korisno osloboditi memoriju poslužitelja pomoću funkcije *mysql_free_result*.

12.4. Upis podatka u bazu podataka

Upisivanje podatka u bazu podataka vrši se pomoću INSERT upita koji se također izvršava pomoću funkcije *mysql_query*.

Izvršavanje upita koji će ubaciti novi redak u tablicu *kontakti* izgledat će ovako:

```
<?php

  $sql = "INSERT INTO kontakti (Id, Ime, Adresa, Grad,
        Email, Spol, Prijatelj) VALUES ( 2, 'Tom Tomić',
       'Aleja jorgovana 5', 2, 'ttomic@srce.hr', 'M', 'Da')";

   mysql_query($sql);

?>
```

Za provjeru uspješnosti upita, može se dodati provjera vrijednosti koju vraća funkcija mysql_query.

Najčešći uzrok neuspjeha upita bit će pogrešno napisana SQL naredba.

```
<?php
  if (mysql_query($sql))
  {
    echo "Redak je dodan.";
  }
  else
  {
    echo "Upit nije uspješno obavljen!";
  }
?>
```

12.5. Promjena podatka u bazi podataka

Promjena podataka u bazi podataka postiže se pomoću UPDATE upita. Za izvršavanje ove vrste upita također se koristi funkcija *mysql_query*.

Primjer izmjene nad određenim retkom u bazi slijedi:

```
<?php

 mysql_query("UPDATE kontakti SET Spol = 'M' WHERE Id = 2");
?>
```

12.6. Brisanje podatka iz baze podataka

Za brisanje podataka iz baze podataka služe DELETE upiti koji se također izvršavaju pomoću funkcije *mysql_query*.

Primjer brisanja retka koji zadovoljava određeni uvjet izgledat će ovako:

```
<?php

  mysql_query("DELETE from kontakti WHERE Id = 2");

?>
```

12.7. Funkcije *stripSlashes* i *addSlashes*

Znakovi kao što su \, ', " i još neki mogu izazvati grešku u SQL upitu, zbog njihovog posebnog značenja u jeziku SQL.

Ovi se znakovi mogu naći u upitu ako ih korisnik preda zajedno s ulaznim podacima. Moguće je da korisnik pokuša upisati ime koje sadrži apostrof (npr. O'Brien). Također njihovom manipulacijom može se izmijeniti upit pa je moguće da ih zlonamjerni korisnik upiše u pokušaju da izbriše postojeće podatke ili da dođe do podataka koje ne bi smio vidjeti.

Zbog toga je preporučljivo ukloniti te znakove iz svih podataka koji se upisuju u bazu, a upisanao ih je korisnik ili su pročitani iz URL-a.

Za uklanjanje (točnije, neutralizaciju) ovih znakova koristi se funkcija addSlashes, koja ispred svakog takvog znaka \, čime se gubi specijalno značenje koje dani znak ima u jeziku SQL. Funkcija *addSlashes* koristi se prilikom zapisivanja u bazu:

```
<?php
 $ime = addSlashes($_POST["ime"]);
 mysql_query("UPDATE kontakti SET ime = $ime WHERE Id = 2");

?>
```

Da bi se korisniku podaci prikazali u istom obliku u kojem ih je upisao prilikom ispisa podataka iz baze, koristi se funkcija *stripSlashes* koja miče dodane znakove \.

```
<?php

  $rezultat = mysql_query("SELECT ime FROM kontakti");
  while ($redak = mysql_fetch_array($rezultat))
  {
     echo "Ime: " . stripSlashes($redak['ime']) . "<br />";
  }

?>
```

Vježba 12.1 – Povezivanje s bazom i ispis podataka iz baze

Ova vježba je nastavak vježbi 10.1 i 11.1 iz prethodnih poglavlja.

Adresar iz prethodnih vježbi bit će izmijenjen tako da se za spremanje podataka ne koristi datoteka, već baza podataka.

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *OtvoriVezu.php*. U datoteku upišite ovaj kôd:

```
<?php

  function OtvoriVezu()
  {
    $veza = mysql_connect("localhost", "root", "");
    if ($veza)
    {
      if (mysql_select_db("adresar", $veza))
      {
        return $veza;
      }
      else
      {
        exit("Baza s tim imenom ne postoji!");
      }
    }
    else
    {
      exit("Ne mogu se spojiti na MySql server!");
    }
  }
?>
```

Svrha navedene funkcije je stvaranje veze na *MySql* poslužitelj i potom odabir baze podataka "adresar". Ako jedan od koraka ne uspije, izvršavanje skripte će se prekinuti uz poruku o pogrešci za što se koristi funkcija *exit*. Ako stvaranje veze i odabir baze podataka uspiju, kao rezultat se vraća identifikator veze (varijabla *$veza*).

Datoteku spremite u mapu "…\D350\vjezbe\vjezba12.1".

2. U programu *PHP Designer* otvorite datoteku *PregledAdresa.php*. Sve naredbe unutar oznaka za PHP kôd pobrišite i upišite ove tri naredbe:

```
<?php

  include("OtvoriVezu.php");

  $veza = OtvoriVezu();

  $rezultat = mysql_query("SELECT * FROM kontakti");
```

Prva naredba je naredba *include* kojom se unutar kôda ove skripte uključuje i sadržaj datoteke *OtvoriVezu.php* čime funkcija *OtvoriVezu* postaje dostupna za uporabu.

Druga naredba pozvat će funkciju *OtvoriVezu* nakon čega će veza s bazom podataka biti otvorena.

Treća naredba izvršit će upit koji vraća sve retke iz tablice *kontakti*, a rezultat upita bit će dostupan preko varijable *$rezultat*.

3. Nakon ove tri naredbe dodajte sljedeći kôd:

```
      while ($redak = mysql_fetch_array($rezultat))
      {
        $id = $redak['Id'];
        $ime = $redak['Ime'];
        $adresa = $redak['Adresa'];
        $grad = $redak['Grad'];
        $email = $redak['Email'];
        $spol = $redak['Spol'];
        $prijatelj = $redak['Prijatelj'];

        echo "<tr>";
        echo "<td>$ime</td>";
        echo "<td>$adresa</td>";
        echo "<td>$grad</td>";
        echo "<td><a href='UnosPoruke.php?email=$email'>
              $email</a></td>";
        echo "<td>$spol</td>";
        echo "<td>$prijatelj</td>";
        echo "<td><img src='Slike/$ime.jpg'></td>";
        echo "</tr>";
      }
```

Pomoću funkcije *mysql_fetch_array*, u petlji će se jedan po jedan dohvaćati reci koje je upit vratio. Podaci za jedan redak bit će dohvaćani preko imena polja u bazi podataka i spremljeni u varijable. Te varijable će odmah potom biti korištene za ispis jednog retka unutar HTML tablice pomoću naredbe *echo*.

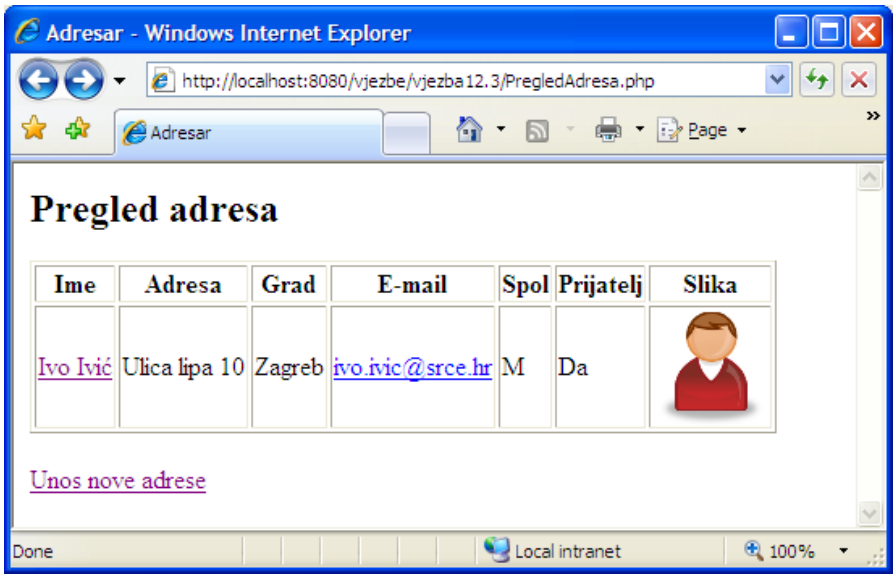
4. Nakon ove petlje dodajte još i naredbe za oslobođenje rezultata i zatvaranje veze na bazu podataka:

```
    mysql_free_result($rezultat);
    mysql_close($veza);
?>
```

5. Spremite datoteku.

6. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

7. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba12.1/PregledAdresa.php* .



Primijetite da je ispisan redak koji je unesen u bazu podataka *adresar* preko aplikacije *phpMyAdmin* u prethodnoj vježbi.

Vježba 12.2 – Ispis podatka koji zadovoljava uvjet

Na stranici za pregled adresa dodaje se link na stranicu koja će prikazati samo podatke za odabrani kontakt, i to unutar obrasca, da bi se podaci kasnije mogli izmijeniti. Obrazac će biti sličan kao obrazac za unos adrese, ali neće biti prazan, već će unutar njegovih elemenata biti prikazani podaci.

Ova vježba je nastavak vježbe 12.1.

1. Otvorite datoteku *PregledAdresa.php* u mapi *localhost/vjezbe/vjezba12.2* i u njoj napravite sljedeću izmjenu:

```
echo "<tr>";
echo "<td><a href='IzmjenaAdrese.php?id=$id'>$ime</a></td>";
echo "<td>$adresa</td>";
echo "<td>$grad</td>";
```

Ime osobe će biti link koji će voditi na stranicu *IzmjenaAdrese.php*, a parametar koji će se proslijediti toj stranici putem URL-a bit će *Id* kontakta u bazi.

Spremite izmijenjenu datoteku.

2. U programu *PHP Designer* stvorite novu datoteku *IzmjenaAdrese.php*. Ova će datoteka sadržavati obrazac u kojem će se prikazati podaci za odabrani kontakt (čija vrijednost polja *Id* je proslijeđena u URL-u), a ti podaci će se moći i promijeniti.

3. Unutar stvorene datoteke dodajte oznaku *head*, a zatim oznake *body* i *h2*:

```
<html>
<head>
  <title>Adresar</title>
  <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
</head>
<body>
  <h2>Izmjena adrese</h2>

</body>
</html>
```

4. Nakon oznake *h2* ubacite oznake za PHP kôd, i sljedeće naredbe unutar njih:

```
<?php
  include("OtvoriVezu.php");
  $veza = OtvoriVezu();
  $rezultat = mysql_query(
    "SELECT * FROM kontakti WHERE Id = " . $_GET['id'] );
  if($redak = mysql_fetch_array($rezultat))
  {
     $id = $redak['Id'];
     $ime = $redak['Ime'];
     $adresa = $redak['Adresa'];
     $grad = $redak['Grad'];
     $email = $redak['Email'];
     $spol = $redak['Spol'];
     $prijatelj = $redak['Prijatelj'];
  }
  mysql_free_result($rezultat);
  mysql_close($veza);
?>
```

Najprije se otvara veza na bazu podataka, te se zatim izvršava *SELECT* upit koji će vratiti redak s čija vrijednost polja *Id* bila proslijeđena u URL-u (i pročitana iz polja *$_GET*).

Zatim se podaci iz pročitanog retka spremaju u varijable, nakon čega se oslobađa identifikator rezultata i zatvara veza na bazu.

5. Poslije PHP oznaka dodajte oznaku za obrazac (*form*):

```
  ?>
  <form method="POST" action="SpremiIzmjene.php">

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

Obrazac će koristiti metodu *POST* za slanje podataka, a skripta koja će obrađivati poslane podatke bit će *SpremiIzmjene.php*.

6. Unutar obrasca dodajte dva elementa za unos:

```
<form method="POST" action="SpremiIzmjene.php">
  Ime:<br/>
  <input type="text" name="ime" value="<?php echo $ime ?>">
  <br/><br/>
  Adresa:<br/>
  <textarea name="adresa"><?php echo $adresa ?></textarea>
  <br/><br/>
</form>
```

Tekst unutar polja za unos postavlja se na vrijednosti varijabli *$ime* i *$adresa* (koje sadrže vrijednosti pročitane iz baze u prethodnom PHP bloku naredbi). Tekst se zapisuje u polja postavljanjem vrijednosti atributa *value* (*input* element), odnosno ispisivanjem između početne i završne oznake (*textarea*).

7. Zatim u obrazac dodajte listu za odabir grada:

```
 <br/><br/>
 Grad:<br/>
 <select name="grad">
  <option <?php if ($grad == "Zagreb") echo "selected" ?> >
    Zagreb
  </option>
  <option <?php if ($grad == "Split") echo "selected" ?> >
    Split
  </option>
 </select>
 <br/><br/>
</form>
```

Kod listi za odabir, odabrani element ima postavljen atribut *selected*, pa je unutar svakog *option* elementa potrebno ubaciti PHP kôd koji će ispisati "selected" ako je taj grad jednak gradu pročitanom iz baze.

8. Unutar obrasca dodajte i polje za unos teksta koje će prikazati adresu elektroničke pošte za odabrani kontakt:

```
  <br/><br/>
  E-mail adresa:<br/>
  <input type="text" name="email"
         value="<?php echo $email ?>" >
  <br/><br/>
</form>
```

9. Zatim dodajte dugme za odabir pomoću kojeg će biti prikazan spol odabranog kontakta:

```
  <br/><br/>
  Spol:<br/>
  <input type="radio" value="M" name="spol"
         <?php if ($spol == "M") echo "checked" ?> />
  muški<br/>
  <input type="radio" value="Ž" name="spol"
         <?php if ($spol == "Ž") echo "checked" ?> />
  ženski
  <br/><br/>
</form>
```

Slično kao kod liste za odabir grada, dugme za odabir će biti označeno ako ima atribut *checked*. PHP kôd ubačen unutar HTML oznake za dugme ispisat će "checked" ako vrijednost odgovara vrijednosti iz baze.

10. Na isti način rješava se i kućica za označavanje koja govori je li odabrani kontakt prijatelj ili ne. Kućica za označavanje također ima atribut *checked* ako je označena.

```
  <br/><br/>
  Prijatelj:<br/>
  <input type="checkbox" name="prijatelj"
         <?php if ($prijatelj == "Da") echo "checked" ?> />
  <br/><br/>
</form>
```

11. Unutar obrasca dodajte i skriveno polje, preko kojeg će se vrijednost polja *Id* proslijediti skripti *SpremiIzmjenu.php*, koja će spremiti izmijenjene podatke u bazu podataka.

```
  <br/><br/>
  <input type="hidden" name="id" value="<?php echo $id ?>">
</form>
```

12. Na kraju, dodajte i dugme za slanje podataka.

```
   <input type="submit" value="Spremi promjene"/>
</form>
```

13. Dodat ćemo još i link za povratak na stranicu s pregledom adresa:

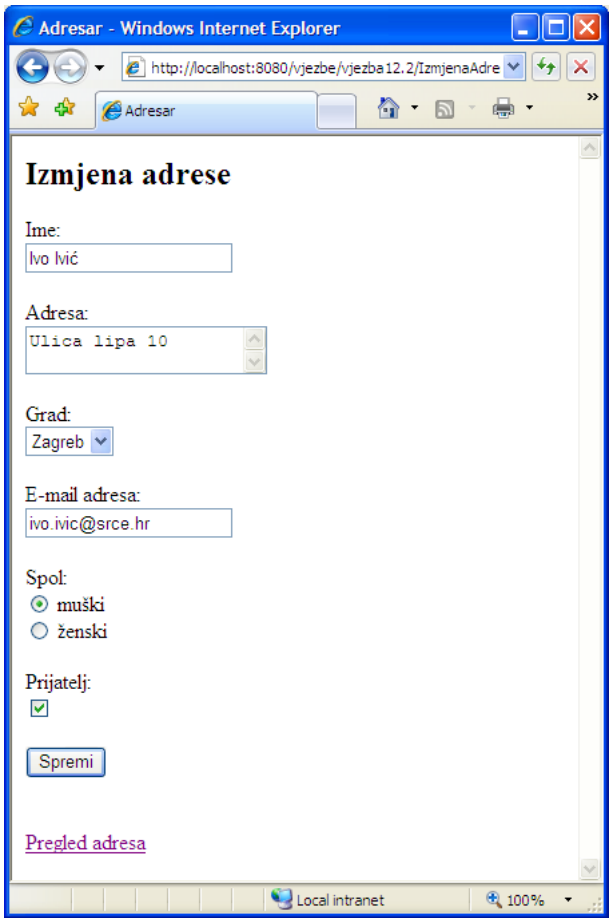
```
 </form>
 <br/>
 <a href="PregledAdresa.php">Pregled adresa</a>
</body>
```

Datoteku *IzmjenaAdrese.php* spremite u mapu "…\D350\vjezbe\vjezba12.2".

8. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

9. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba12.2/PregledAdresa.php* .

10. Kliknite na ime prikazanog kontakta.



Link nas je odveo na stranicu *IzmjenaAdrese.php?id=1*. U obrascu su prikazani podaci za redak iz tablice *kontakti* koji ima vrijednost polja *Id* jednaku 1.

Vježba 12.3 – Izmjena podatka u bazi podataka

Ova vježba je nastavak vježbe 12.2.

Izmjene unesene u obrazac iz prethodne vježbe bit će spremljene u bazu podataka preko skripte *SpremiIzmjenu.php*.

1. U programu *PHP Designer* stvorite novu datoteku *SpremiIzmjenu.php*.
2. Unutar stvorene datoteke dodajte oznaku *head*, a zatim oznake *body* i *h2*:

```
<html>
<head>
  <title>Adresar</title>
  <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
</head>
<body>
  <h2>Izmjena adrese</h2>
</body>
</html>
```

3. Nakon oznake *h2* ubacite oznake za PHP kôd i sljedeće naredbe unutar njih:

```
<?php

  include("OtvoriVezu.php");

  $id = $_POST['id'];
  $ime = $_POST['ime'];
  $adresa = $_POST['adresa'];
  $grad = $_POST['grad'];
  $email= $_POST['email'];
  $spol = $_POST['spol'];

  if (isset($_POST['prijatelj']))
      $prijatelj = "Da";
  else
      $prijatelj = "Ne";
?>
```

Prva naredba uključuje sadržaj datoteke *OtvoriVezu.php* u dostupni kôd, kako bi funkcija *OtvoriVezu* bila dostupna. Zatim slijedi blok naredbi koji čitaju podatke proslijeđene s obrasca metodom *POST*. Ako kućica za označavanje nije bila označena, u polju *$_POST* neće biti dostupan element pod nazivom "prijatelj" (što provjeravamo pomoću funkcije *isset*) pa stoga postavljamo vrijednost varijable *$prijatelj* na "Ne".

4. Nakon ovih naredbi, još uvijek unutar oznaka za PHP kôd, dodajte sljedeće naredbe:

```
 $sql = "UPDATE kontakti SET Ime = '$ime', Adresa = '$adresa'";
 $sql .= ", Grad = '$grad', Email='$email', Spol = '$spol'";
 $sql .= ", Prijatelj='$prijatelj' WHERE Id = $id";

 $veza = OtvoriVezu();
 if (mysql_query($sql))
   echo "Izmjena je uspješno spremljena.";
 else
   echo "Izmjena podataka u bazi nije uspjela.";

 mysql_close($veza);

?>
```

Prve tri naredbe služe za slaganje teksta UPDATE upita kojim će se izmijeniti postojeći podaci u bazi podataka. Za odgovarajući redak, polja će biti izmijenjena u vrijednosti koje su primljene iz obrasca.

Zatim se otvara veza s bazom podataka, te se izvršava upit pomoću funkcije *mysql_query*. U slučaju uspješnog ili neuspješnog izvršavanja upita, ispisuje se odgovarajuća poruka. Nakon toga, zatvara se veza s bazom podataka.

5. Nakon bloka PHP kôda dodajte link za povratak na pregled adresa.

```
  ?>
  <br /><br />
  <a href="PregledAdresa.php">Pregled adresa</a>
</body>
</html>
```

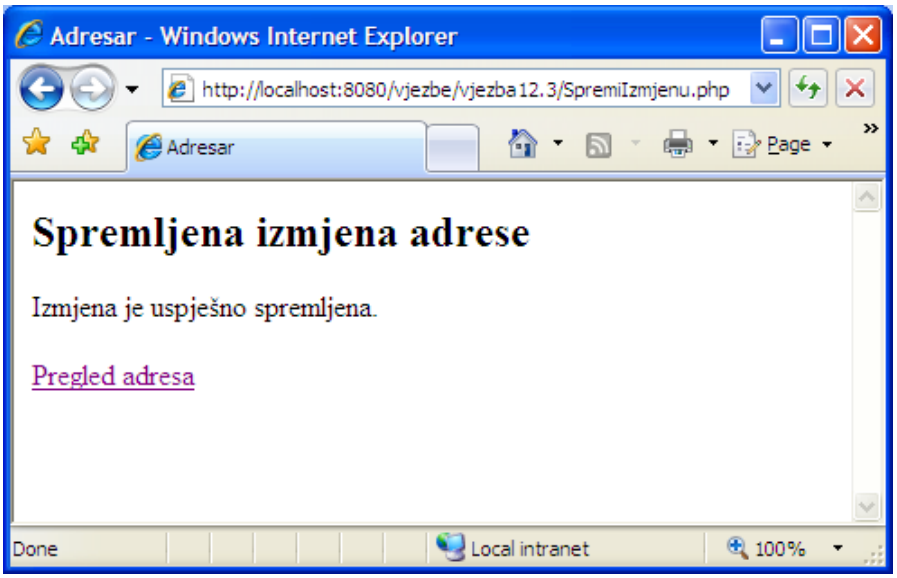
Spremite datoteku u mapu "…\D350\vjezbe\vjezba12.3".

6. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

7. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba12.3/PregledAdresa.php* .

8. Kliknite na ime kontakta.

9. Izmijenite prikazane podatke i kliknite na "Spremi".



10. Nakon što se prikaže poruka da je izmjena spremljena, kliknite na link "Pregled adresa". Uočavate da su na pregledu adresa prikazani izmijenjeni podaci.

11. Pokrenite aplikaciju phpMyAdmin. (U web preglednik upišite adresu: *http://localhost/mysql/*. Uvjerite se da su podaci zaista promijenjeni u bazi.

Vježba 12.4 – Upis podatka u bazu podataka

Ova vježba je nastavak vježbe 12.3.

U ovoj vježbi se mijenja skripta *SpremiAdresu.php* tako da se novouneseni kontakt sprema u bazu podataka a ne u tekstualnu datoteku.

1. U programu *PHP Designer* otvorite datoteku *SpremiAdresu.php* u mapi "…\D350\vjezbe\vjezba12.4".

2. Na početak bloka PHP naredbi dodajte naredbu *include* koja će u skriptu uključiti kôd datoteke *OtvoriVezu.php*.

```
<?php

  include("OtvoriVezu.php");

  $ime = $_POST['ime'];
```

3. Obrišite sljedeće naredbe koje služe za zapisivanje podataka u datoteku:

```
  // spremi vrijednosti u datoteku
  $datoteka = fopen("Adresar.txt", 'a');
  fwrite($datoteka,
         "\n$ime\n$adresa\n$grad\n$email\n$spol\n$prijatelj");
  fclose($datoteka);
```

Umjesto njih upišite ove naredbe:

```
  $sql = "INSERT INTO kontakti ";
  $sql .= "(Ime, Adresa, Grad, Email, Spol, Prijatelj) ";
  $sql .= "VALUES ( '$ime', '$adresa', '$grad', ";
  $sql .= "'$email', '$spol', '$prijatelj' )";

  $veza = OtvoriVezu();

  if (mysql_query($sql))
    echo "Podaci su uspješno spremljeni.";
  else
    echo "Spremanje podataka u bazu nije uspjelo.";

  mysql_close($veza);
```

Najprije se unosi INSERT upit pomoću kojega će kontakt biti ubačen u bazu podataka. Primjećujete da u njemu nije potrebno navoditi vrijednost polja *Id* (koje je u bazi podataka podešeno da se automatski povećava za 1, pomoću opcije *auto_increment*).

Nakon toga se otvara veza s bazom. Zatim se izvršava upit, te se prikazuje poruka o uspjehu upita. Na kraju se veza s bazom zatvara.

4. Promijenite naredbu koja sprema sliku, tako da se sada za naziv slike upotrebljava identifikator iz baze, a ne ime osobe:

```
// spremi sliku
if ($_FILES["slika"]["tmp_name"] &&
    $_FILES["slika"]["size"] != 0)
{
    move_uploaded_file($_FILES["slika"]["tmp_name"],
      "Slike/$ime.jpg");
}
```

5. Spremite datoteku.

6. Pokrenite servis *EasyPHP* ako već nije pokrenut.

7. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba12.4/UnosAdrese.htm* .

8. Unesite novu adresu i kliknite na "Spremi".

9. Nakon potvrde o uspješno spremljenim podacima kliknite na link "Pregled adresa" i uvjerite se da je novi unos prikazan u listi.

Vježba 12.5 – Brisanje podatka iz baze podataka

Ova vježba je nastavak vježbe 12.4.

Na stranicu s pregledom adresa potrebno je dodati link za brisanje pojedinog kontakta, i skriptu koja će obaviti brisanje.

1. U programu *PHP Designer* otvorite datoteku *PregledAdresa.php* u mapi "…\D350\vjezbe\vjezba12.5".

2. Dodajte još jednu ćeliju unutar zaglavlja HTML tablice:

```
    <th>Slika</th>
    <th>Brisanje</th>
  </tr>
```

3. Unutar PHP kôda koji ispisuje redak s podacima, dodajte naredbu koja će ispisati ćeliju s linkom na skriptu za brisanje:

```
   echo "<td><img src='Slike/$ime.jpg'></td>";
   echo "<td>
        <a href='ObrisiAdresu.php?id=$id'>Obriši</a></td>";
   echo "</tr>";
```

Skripti *ObrisiAdresu.php* bit će, putem URL-a, proslijeđen *id* kontakta koji treba obrisati.

(Napomena: naredbu treba napisati u jednom retku.)

Spremite datoteku.

4. U programu *PHP Designer* stvorite novu datoteku *ObrisiAdresu.php*.

5. Unutar stvorene datoteke dodajte oznaku *head*, a zatim oznake *body* i *h2*:

```
<html>
<head>
  <title>Adresar</title>
  <meta http-equiv="Content-Type" content="text/html;
   charset=windows-1250" />
</head>
<body>
  <h2>Brisanje adrese</h2>

</body>
</html>
```

6. Nakon oznake *h2* ubacite oznake za PHP kôd i sljedeće naredbe unutar njih:

```
<?php
  include("OtvoriVezu.php");
  $id = $_GET['id'];
  $veza = OtvoriVezu();
  if (mysql_query("DELETE FROM kontakti WHERE Id = $id"))
  {
    echo "Zapis je obrisan.";
  }
  mysql_close($veza);
?>
```

Prva naredba uključuje sadržaj datoteke *OtvoriVezu.php* u dostupni kôd, kako bi funkcija *OtvoriVezu* bila dostupna. Zatim se iz polja *$_GET* čita *Id* kontakta, prosljeđen putem URL-a sa stranice *PregledAdresa.php*, i sprema u varijablu *$id*.

Sljedeća naredba otvara vezu s bazom podataka, a nakon toga se izvršava upit koji će obrisati redak sa zadanim *Id*-om. Ako je brisanje uspjelo, ispisat će se poruka o tome. Posljednja naredba zatvorit će vezu s bazom podataka.

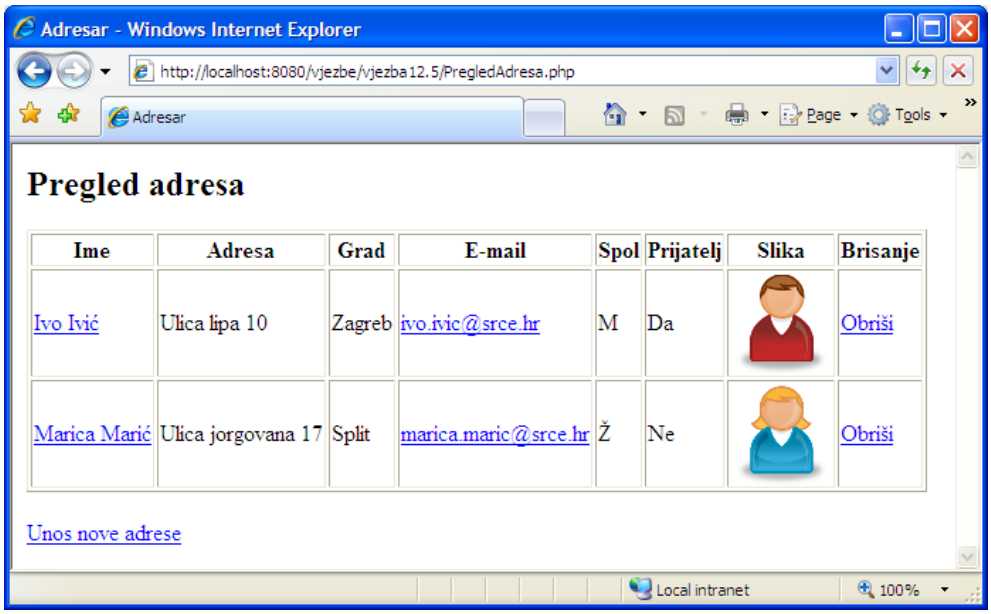
7. Nakon bloka PHP kôda potrebno je dodati još i link za povratak na pregled adresa.

```
  ?>
  <br /><br />
  <a href="PregledAdresa.php">Pregled adresa</a>
</body>
</html>
```

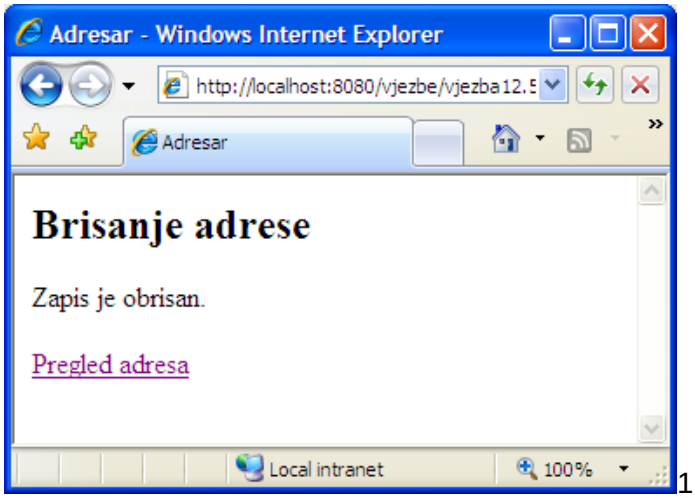
Spremite datoteku u mapu "…\D350\vjezbe\vjezba12.5".

8. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

9. U web preglednik upišite adresu:
   http://localhost/vjezbe/vjezba12.5/PregledAdresa.php .



10. Kliknite na jedan od linkova za brisanje.



1

11. Nakon što se prikaže poruka da je brisanje uspješno obavljeno, vratite se na pregled adresa i uočite da obrisanog kontakta više nema u popisu.

U ovom je poglavlju obrađeno:

- povezivanje s bazom bodataka MySQL – funkcija ***mysql_connect***
- odabir baze - funkcija ***mysql_select_db***
- zatvaranje veze - funkcija ***mysql_close***
- izvršavanje SQL upita - funkcija ***mysql_query***
- dohvaćanje rezultata SELECT upita – funkcije ***mysql_fetch_array*** i ***mysql_fetch_row***
- dobivanje broja redaka – funkcija ***mysql_num_rows***

13. Sjednice i autentikacija korisnika

Jedan od nedostataka HTTP protokola je taj što se između dva zahtjeva ne čuva trenutno stanje. Kad stigne novi zahtjev, poslužitelj ne zna da je on stigao od istog korisnika kao i prethodni zahtjev i da se nastavlja na prethodne korisnikove akcije.

Da bi se moglo pratiti pojedinog korisnika, za svakog korisnika se stvara sjednica. Sjednica započinje spajanjem korisnika (klijenta) na poslužitelj, a završava njegovim odlaskom (nakon što protekne određeno vrijeme, najčešće 30 minuta od njegove zadnje akcije).

13.1. Kolačići

Jedan od načina za ostvarivanje sjednica je korištenje *kolačića*. ***Kolačić*** je kratak niz tekstualnih podataka koji razmjenjuju poslužitelj i klijent unutar zaglavlja HTTP zahtjeva i odgovora.

Razmjena podataka pomoću *kolačića* teče ovako:

1. klijent šalje zahtjev poslužitelju za nekom stranicom
2. poslužitelj šalje odgovor i zajedno s njim *kolačić*
3. klijent sprema *kolačić* (kao tekstualnu datoteku na korisnikovom računalu)
4. pri sljedećem zahtjevu poslanom na isti poslužitelj, klijent šalje *kolačić* natrag na poslužitelj

Kolačići se često koriste tako da se unutar *kolačića* zapisuje identifikator koji dobiva korisnik (odnosno identifikator sjednice između klijenta i poslužitelja). Na taj način, kad klijent u sljedećem zahtjevu pošalje isti identifikator natrag, poslužitelj zna da je riječ o istom korisniku.

Kolačići se također koriste i za spremanje raznih korisničkih postavki (npr. odabir jezika stranica koje će se pretraživati kod tražilice Google), tako da se sljedeći put kad korisnik posjeti istu stranicu, primjenjuju postavke koje je odabrao prošli put.

Za slanje kolačića u PHP-u koristi se funkcija ***setcookie***. Argumenti koje ona prima su:

- naziv *kolačića* (znakovni niz)
- vrijednost (znakovni niz; opcionalno)
- datum do kada *kolačić* vrijedi (Unixova vremenska oznaka; opcionalno) – nakon tog datuma *kolačić* će biti izbrisan s korisnikovog računala
- putanja (znakovni niz; opcionalno) – putanja (dio URL-a nakon domene) za koju vrijedi *kolačić*
- domena (znakovni niz; opcionalno) – internetska domena poslužitelja; *kolačić* će biti poslan natrag ako domena i putanja odgovaraju URL-u na koji se šalje zahtjev

HTTPS veza je veza između poslužitelja i klijenta koja se odvija putem HTTPS protokola. Radi se o nadgradnji HTTP protokola (dodavanjem SSL sloja – *Secure Sockets Layer*) u kojoj se podaci ne šalju u jasnom (*clear text*), već u enkriptiranom obliku.

- sigurnost (cijeli broj; opcionalno) – ako je ovaj argument 1, *kolačić* će se slati samo preko sigurne (HTTPS) veze

Primjer slanja *kolačića* pomoću funkcije *setcookie*:

```
<?php
  setcookie("korisnik", "Ivica Ivić");
?>
```

Pri slanju zahtjeva na poslužitelj, klijent će poslati i sve *kolačiće* čija domena i putanja odgovaraju URL-u poslužitelja. *Kolačićima* koje klijent šalje poslužitelj može pristupiti preko globalno dostupnog polja *$_COOKIE*.

Ovako bi poslužitelj pristupio *kolačiću* čiji je naziv "korisnik":

```
<?php
  $kupac = $_COOKIE["korisnik"];
  echo $kupac;
?>
```

```
Ivica Ivić
```

13.2. Sjednice

Poslužitelj ostvaruje **sjednicu** (spoj, sesiju) s pojedinim klijentom kao niz HTTP zahtjeva i odgovora između njih. Svaka sjednica dobiva svoj jedinstveni identifikator, na temelju kojega se prepoznaje da pojedini zahtjevi pripadaju točno određenoj sjednici.

Identifikator sjednice se stvara na poslužitelju, i šalje klijentu unutar HTTP odgovora. Klijent ga potom prosljeđuje natrag prilikom svakog sljedećeg zahtjeva.

Sjednica će trajati sve dok stižu zahtjevi klijenta. Ako od posljednjeg zahtjeva protekne određeno vrijeme (u PHP-u je predefinirana vrijednost tog vremena 30 minuta), sjednica će se automatski zatvoriti.

Osnovni način za prosljeđivanje identifikatora sjednice i njegovo čuvanje na klijentskom računalu su kolačići. Ako su kolačići onemogućeni ili nisu podržani u korisnikovom *web* pregledniku, identifikator sjednice će se prosljeđivati preko URL-a. Primjer takvog URL-a:

http://www.srce.hr?PHPSESSID=242c489fb4a1bc79f5cf365988167e4d

Sjednice se mogu upotrijebiti za spremanje podataka koji su vezani za sjednicu, dakle za pojedinog korisnika. Takvi podaci spremaju se u varijable sjednice koje će biti vidljive samo za tog korisnika i trajat će samo dok traje i sjednica.

Vrijednosti varijabli sjednice spremaju se, ako nije drugačije postavljeno, u datotekama u mapi */tmp* na poslužitelju.

Varijable sjednice

Varijablama sjednica pristupa se preko globalno dostupnog polja *$_SESSION*. Stvaranje nove varijable sjednice je zapravo dodavanje novog člana u polje *$_SESSION*:

```
<?php
  $_SESSION["korisnik"] = $imeKorisnika;
?>
```

Čitanje vrijednosti varijable sjednice je pristupanje članu polja *$_SESSION* pomoću odgovarajućeg znakovnog ključa:

```
<?php
  if (isset($_SESSION["korisnik"]))
  {
    echo $_SESSION["korisnik"];
   }
?>
```

Za provjeru je li neka varijabla sjednice stvorena ili ne, može se koristiti funkcija *isset*.

Varijable sjednice čuvaju se na poslužitelju. Preko identifikatora sjednice, moguće je pristupiti varijablama koje pripadaju odgovarajućoj sjednici. Varijable sjednice će za svakog korisnika sadržavati različite vrijednosti.

Uz podatke poslane unutar URL-a (metodom *GET*) ili preko obrasca (metodom *POST*), korištenje varijabli sjednice je još jedan način čuvanja podataka između dva HTTP zahtjeva.

Otvaranje sjednice

Za započinjanje sjednice, potrebno je pozvati funkciju ***session_start***. Funkcija ne prima nijedan argument i uvijek vraća vrijednost *TRUE*.

Čak i ako je sjednica već započeta, funkciju *session_start* potrebno je pozvati na početku svake skripte koja koristi varijable sjednice, jer se, kad se ona pozove, varijable sjednice učitavaju u polje *$_SESSION*.

Primjer poziva funkcije:

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

Ako se na poslužitelju vrijednost *session.auto_start* postavi na 1, sjednica će automatski započinjati po primanju prvog klijentovog zahtjeva. Predefinirana vrijednost ove postavke je 0.

Zatvaranje sjednice

Sjednica će se automatski zatvoriti nakon što protekne zadani vremenski rok od posljednjeg zahtjeva klijenta. No, sjednicu je moguće i prije zatvoriti, upotrebom funkcije ***session_destroy***. Ova funkcija također ne prima nijedan argument i uvijek vraća vrijednost *TRUE*:

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

Nakon poziva ove funkcije, vrijednosti svih varijabli sjednice bit će izgubljene.

13.3. Autentikacija korisnika

Autentikacija je provjera je li korisnik onaj za koga se predstavlja, dok je **autorizacija** provjera ima li korisnik (čiji je identitet već potvrđen) prava za pristup određenom resursu.

Autentikacija je provjera identiteta korisnika, odnosno potvrđivanje da je on doista onaj za koga se predstavlja. Najčešći način ostvarivanja autentikacije je pomoću korisničkog imena i lozinke. Korisnik se predstavlja svojim korisničkim imenom, dok se provjera je li to stvarno on izvršava pomoću lozinke.

Pri prvom pristupanju korisnika aplikaciji, od korisnika se traži unos korisničkog imena i lozinke. Nakon toga se provjerava postoji li korisnik s tim korisničkim imenom i tom lozinkom u bazi podataka. Ako ne postoji, pristup ostatku aplikacije mu se onemogućuje dok ne upiše točno ime i lozinku. (Dozvoljeni broj pokušaja upisivanja imena i lozinke se često ograničava.)

Jedna od mogućnosti za neovlašteni pristup aplikaciji je tzv. *session hijacking*. Ako napadač presretne klijentov HTTP zahtjev, iz njega može preuzeti identifikator sjednice i predstaviti se kao taj klijent. No, ista opasnost postoji i ako napadač presretne HTTP zahtjev koji sadrži lozinku upisanu u obrazac. Zbog toga se u slučajevima kad je potrebna veća sigurnost koristi HTTPS protokol.

Ako je korisnik upisao odgovarajuće ime i lozinku, omogućit će mu se pristup ostatku aplikacije. Na ovom mjestu će se najčešće, korištenjem varijable sjednice, zapisati da je autentikacija korisnika izvršena. Zahvaljujući tome, korisnik neće morati ponovno upisivati lozinku prilikom svakog novog zahtjeva.

Kad sjednica istekne, korisnik će se morati ponovno autenticirati upisivanjem korisničkog imena i lozinke.

Postavlja se pitanje je li moguće da korisnik podastre lažni identifikator sjednice (koji može promijeniti izmjenom *kolačića* ili URL-a) i na taj način se predstavi kao već autenticirani korisnik. Ta je mogućnost matematički vrlo malo vjerojatna, zato što svaka PHP sjednica dobiva nasumični identifikator, koji se zatim enkriptira.

I lozinke se u bazi podataka često drže u kriptiranom obliku, da netko tko ima pristup bazi ne bi mogao preuzeti lozinke drugih korisnika.

Za enkripciju lozinki, (kao i identifikatora sjednice), koristi se posebna metoda enkripcije – izračunavanje sažetka poruke (*message digest*, *hash*) koja je jednosmjerna. To znači da se iz zapisa u bazi ne može dobiti originalna lozinka, ali se iz lozinke može lako dobiti njen enkriptirani oblik i tako provjeriti je li upisana lozinka točna.

Vježba 13.1 – Korištenje *kolačića*

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *zapisiKolacice.php*.

2. U datoteci napišite ove naredbe:

```
<?php
  setcookie("kupac", "Ivica Ivić");
  setcookie("proizvod", "cipele");
  setcookie("cijena", "299 kn");
?>
```

   Ove naredbe stvorit će tri *kolačića* i poslati ih klijentu unutar HTTP odgovora.

3. Poslije oznaka za PHP kôd dodajte link na stranicu *procitajKolacice.php*:

```
?>
<a href="procitajKolacice.php">Pročitaj kolačiće</a>
```

4. Spremite datoteku u mapu "..\D350\vjezbe\vjezba13.1" .

5. U programu *PHP Designer* stvorite novu datoteku *procitajKolacice.php*.

6. U datoteci napišite ove naredbe: :

```
<?php
  foreach ($_COOKIE as $cookie)
  {
     echo $cookie . "<br/>";
  }
?>
```

   Skripta *procitajKolacice.php* služit će za čitanje i ispis vrijednosti *kolačića*. U polju *$_COOKIE* nalazit će se svi *kolačići* koje je klijent poslao poslužitelju.

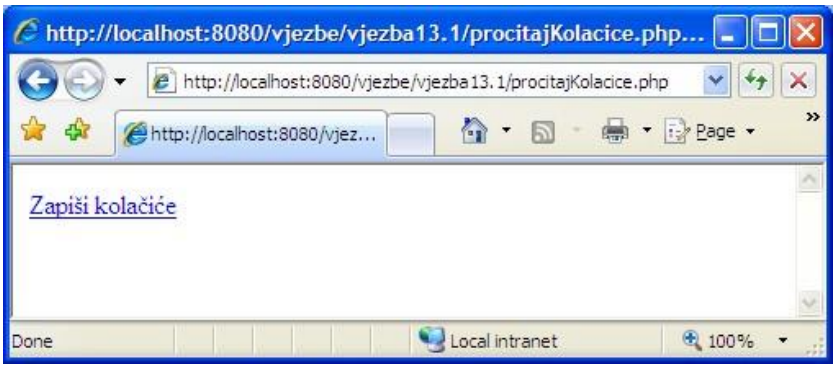
7. Poslije oznaka za PHP kôd dodajte link na stranicu *zapisiKolacice.php*:

```
?>
<a href="zapisiKolacice.php">Zapiši kolačiće</a>
```

8. Spremite datoteku u mapu "..\D350\vjezbe\vjezba13.1" .

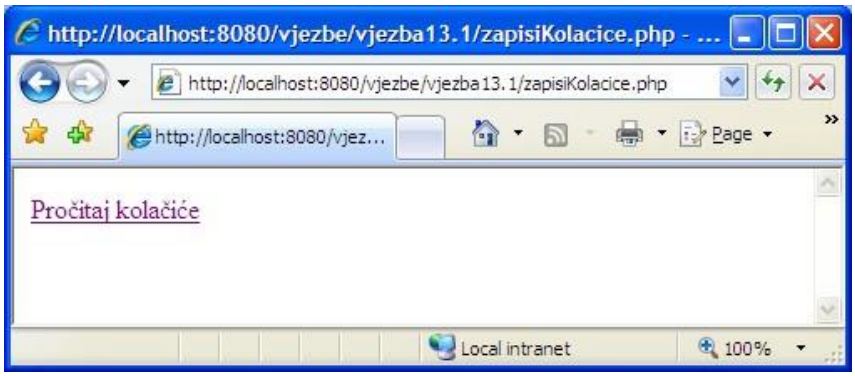
9. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

10. U web preglednik upišite adresu:
    http://localhost/vjezbe/vjezba13.1/procitajKolacice.php .

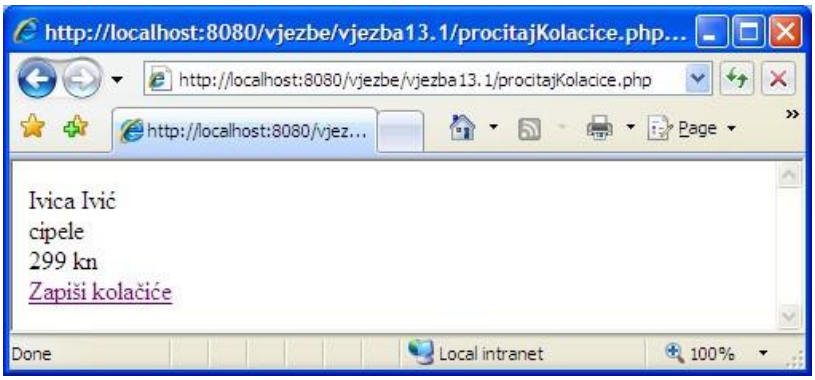


Kolačići još nisu poslani klijentu, pa skripta *procitajKolacice.php* neće pronaći u polju *$_COOKIE* nijedan *kolačić*.

11. Kliknite na link "Zapiši kolačiće". Izvršit će se skripta *zapisiKolacice.php* i klijentu će se poslati tri *kolačića* koja će on zapisati lokalno.



12. Sada kliknite na link "Pročitaj kolačiće". Klijent će poslati zahtjev za skriptom *procitajKolacice.php*, i unutar njega, sve *kolačiće* koje je primio u prethodnom odgovoru.



Ispisat će se svi *kolačići* iz polja *$_COOKIE*, a to će biti svi *kolačići* koje je poslala skripta *zapisiKolacice.php*.

Vježba 13.2 – Sjednice i autentikacija korisnika

1. Pokrenite program *PHP Designer* i stvorite novu datoteku *prijava.php*.
2. U datoteci napišite obrazac koji će služiti za unos korisničkog imena i lozinke:

```
<html>
  <head>
    <title>Prijava na sustav</title>
  </head>
  <body>
    <h2>Prijava na sustav</h2>
    <form method="POST" action="prijava.php">
      <table>
        <tr>
          <td>Korisničko ime</td>
          <td><input type="text" name="korisnickoIme"></td>
        </tr>
        <tr>
          <td>Lozinka</td>
          <td><input type="password" name="lozinka"/></td>
        </tr>
        <tr>
          <td></td>
          <td><input type="submit" value="Prijavi se"/></td>
        </tr>
      </table>
      <?php echo $poruka ?>
    </form>
  </body>
</html>
```

Obrazac će slati podatke pomoću metode *POST*. Budući da su podaci koji se šalju korisničko ime i lozinka, bolje ih je slati metodom *POST* a ne *GET* (kako ne bi bili vidljivi unutar URL-a). Skripta koja će primiti podatke bit će ista ova skripta, *prijava.php*.

3. Prije HTML oznaka umetnite oznake za PHP kôd i ove naredbe unutar njih:

```
<?php
  session_start();
  $poruka = "";

  if ( isset($_POST["korisnickoIme"]) &&
       isset($_POST["lozinka"]))
  {
    $korisnickoIme = $_POST["korisnickoIme"];
    $lozinka = $_POST["lozinka"];

    if ($korisnickoIme == "admin" && $lozinka == "123")
    {
      $_SESSION["korisnickoIme"] = $korisnickoIme;
      header("Location: index.php");
    }
    else
    {
      $poruka = "Neuspješna prijava!";
    }
  }
?>
```

Prva naredba (poziv funkcije *session_start*)započinje sjednicu, ako je nije već započeta. Osim toga, učitava vrijednosti varijabli sjednice u polje *$_SESSION*.

Zatim se provjerava jesu li metodom *POST* poslani korisničko ime i lozinka. Za provjeru jesu li postavljeni pojedini članovi polja *$_POST* koristi se funkcija *isset*. Ako jesu, provjeravaju se njihove vrijednosti. U ovom jednostavnom primjeru bit će prihvaćen samo korisnik s korisničkim imenom "admin" i lozinkom "123". (U stvarnoj primjeni korisničko ime i lozinka držali bi se u bazi podataka.)

Ako su korisnički podaci točni, korisničko ime će se zapisati u varijablu sjednice. Naziv te varijable sjednice, odnosno njen znakovni ključ u polju *$_SESSION*, bit će "korisnickoIme". Potom će, korištenjem funkcije *header* korisnik biti preusmjeren na stranicu *index.php*.

4. Spremite datoteku u mapu "..\D350\vjezbe\vjezba13.2".

5. U programu *PHP Designer* stvorite novu datoteku *index.php*.

6. U datoteku upišite sljedeće naredbe, unutar oznaka za PHP kôd:

```
<?php

  session_start();

  if (!isset($_SESSION["korisnickoIme"]))
  {
    header("Location: prijava.php");
  }

?>
```

I u ovoj datoteci najprije će se, pozivom funkcije *session_start*, započeti sjednica, ako nije već započeta i učitati vrijednosti varijabli sjednice u polje *$_SESSION*.

Nakon toga se provjerava je li postavljena varijabla sjednice s nazivom "korisnickoIme". Ako nije, korisnik će biti preusmjeren na stranicu za unos lozinke (*prijava.php*).

7. Nakon oznaka za PHP kôd napišite ovaj kôd:

```
?>
<html>
  <head>
    <title>Dobrodošli</title>
  </head>
  <body>
    <h2>
      Dobrodošli,
      <?php echo $_SESSION["korisnickoIme"] ?>
    </h2>
    <p>Sada imate neograničeni pristup ovoj stranici.</p>
    <a href="odjava.php">Odjavi se</a>
  </body>
</html>
```

Unutar HTML-a ubačena je PHP naredba koja će ispisati ime korisnika pročitano iz varijable sjednice na taj način, da će biti ispisano "Dobrodošli, "*korisnickoIme*". Na stranici se nalazi i link za odjavljivanje korisnika.

8. Spremite datoteku u mapu "..\D350\vjezbe\vjezba13.2".

9. U programu *PHP Designer* stvorite novu datoteku *odjava.php*. Ova datoteka će izvršiti odjavu korisnika, odnosno zatvaranje njegove sjednice.

10. U datoteku upišite ove naredbe:

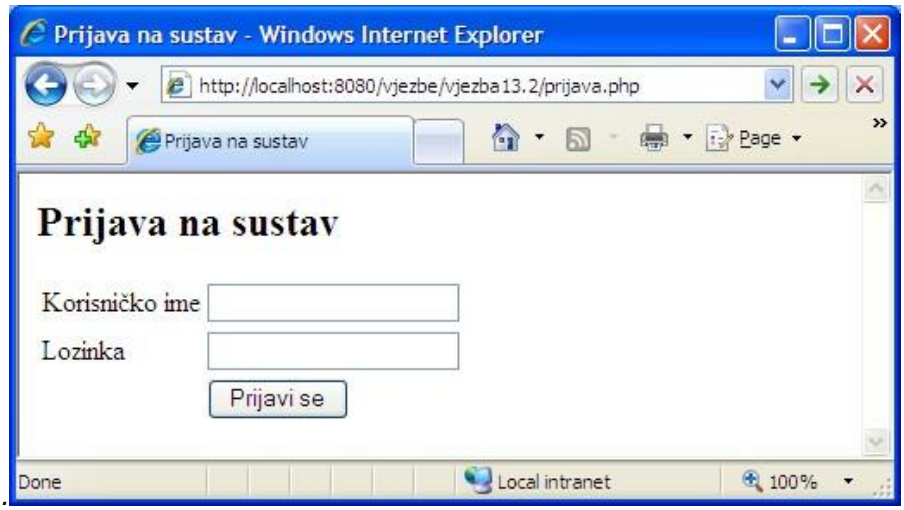
```
<?php

  session_start();
  session_destroy();
  header("Location: index.php");

?>
```

Da bi se sjednicu moglo koristiti u skripti, potrebno je najprije pozvati funkciju *session_start*. Nakon toga poziva se funkcija *session_destroy*, koja će zatvoriti sjednicu i izbrisati sve varijable sjednice. Zatim će se korisnik preusmjeriti na stranicu *index.php*.

11. Spremite datoteku u mapu "..\D350\vjezbe\vjezba13.2".

12. Pokrenite servis *EasyPHP*, ako već nije pokrenut.

13. U web preglednik upišite adresu: *http://localhost/vjezbe/vjezba13.2/* .



Poziva se stranica *index.php*, no kako korisnik još nije prijavljen, preusmjerava ga se na stranicu *prijava.php*.

14. Upišite korisničko ime "admin", lozinku "123" i kliknite na "Prijavi se".

15. Kliknite na "Odjavi se".

    Skripta *odjava.php* zatvorit će sjednicu i preusmjeriti korisnika na stranicu *index.php*. Kako njegovo korisničko ime više nije zapisano u varijabli sjednice, korisnik će biti preusmjeren na stranicu *prijava.php*.

U ovom je poglavlju obrađeno:

- *kolačići* i njihovo korištenje za spremanje malih količina podataka na klijentskom računalu
- sjednica između klijenta i poslužitelja
- načini prenošenja identifikatora sjednice – u kolačiću ili u URL-u
- varijable sjednice
- započinjanje i završavanje sjednice
- autentikacija korisnika pomoću korisničkog imena i lozinke

14. Primjer sustava za upravljanje sadržajem web stranica

Ovo poglavlje se bavi jednim od najčešćih primjera *web* aplikacije – sustavom za upravljanje sadržajem *web* stranica (CMS – *Content Management System*). Sustav će biti ostvaren kao jednostavna aplikacija koja omogućava unos i objavu članaka na *web* sjedištu.

14.1. Zahtjevi sustava

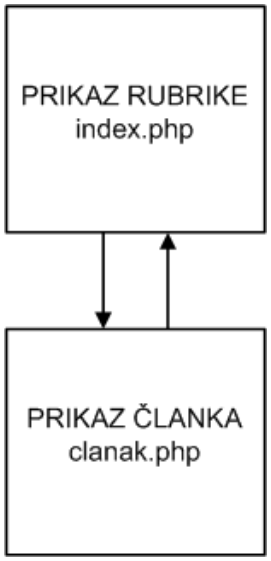
Ovo su osnovni zahtjevi koji se postavljaju za ovakav sustav:

- članci trebaju biti grupirani u rubrike
- uz svaki članak može doći po jedna ili više slika
- sve akcije će biti omogućene samo autenticiranim korisnicima, ostali mogu samo pregledavati rubrike i članke
- treba omogućiti unos članaka, njihovu izmjenu i brisanje
- članak može biti unesen i spremljen, ali neće biti vidljiv na javnim stranicama dok se ne označi kao objavljen
- treba omogućiti vezanje slika uz članak i njihovo brisanje

Da bi sustav bio jednostavniji, i da bi ga se moglo ostvariti u kraćem roku, nije potrebno napraviti sučelja za unos, izmjenu i brisanje rubrika i korisnika. Rad s rubrikama i korisnicima obavljat će se kroz bazu podataka, točnije preko aplikacije *phpMyAdmin*.

14.2. Javni dio sustava

Javni, svima dostupni dio sastojat će se od stranice *index.php* koja prikazuje članke koji pripadaju pojedinoj rubrici, i stranice *clanak.php* koja će prikazivati pojedini članak.



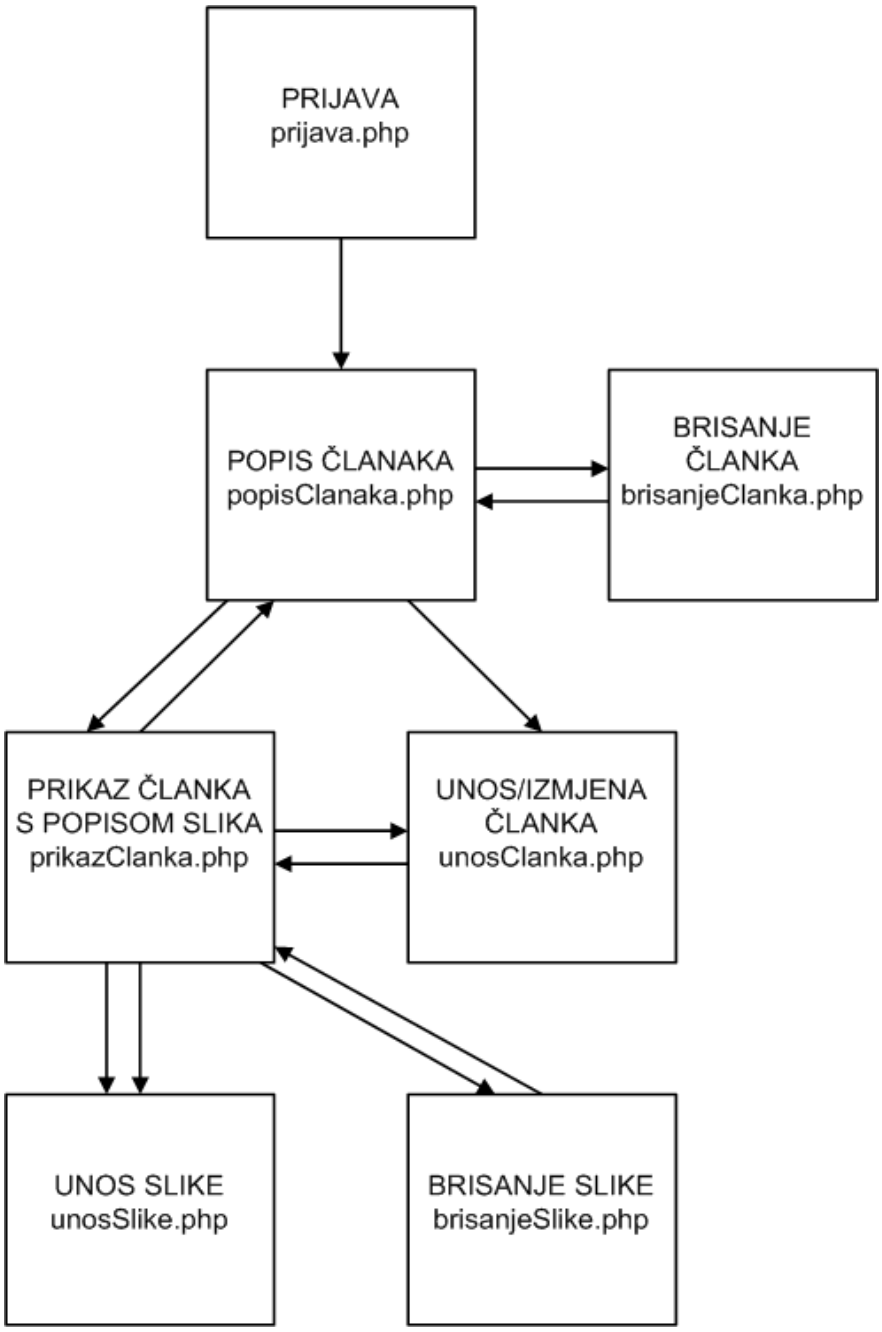
Stranica *index.php* primat će u URL-u identifikator rubrike, a ako se on ne navede, prikazivat će se početna stranica na kojoj će biti posljednjih 5 članaka prema datumu objave.

Stranica *clanak.php* također će u URL-u dobivati identifikator članka.

U lijevom dijelu stranice nalazit će se dio za navigaciju koji će se sastojati od linkova koji će voditi na pojedine rubrike.

14.3. Administrativni dio sustava

Pristup u administrativni dio bit će dopušten samo korisnicima koji prođu autentikaciju. Autentikacija će se obavljati na temelju korisničkog imena i lozinke.



Stranica *prijava.php* služit će za unos korisničkog imena i lozinke kroz obrazac. Obrazac će se slati istoj stranici na kojoj se nalazi – *prijava.php*, nakon ćega će se obaviti provjera postoji li korisnik s unesenim imenom i lozinkom u bazi podataka. Ako postoji, korisnik će biti preusmjeren na stranicu s popisom članaka.

Stranica *popisClanaka.php* bit će početna stranica za administraciju sustava. Na ovoj stranici ispisivat će se popis postojećih članaka u tabličnom obliku. Za svaki članak bit će prikazan link za detaljniji prikaz članka i link za brisanje članka. Nakon popisa članaka bit će prikazan i link za dodavanje novog članka.

Link za brisanje članka vodit će na skriptu *brisanjeClanka.php* koja će obaviti brisanje članka i vratiti korisnika natrag na stranicu s popisom članaka.

Stranica *prikazClanka.php* davat će detaljni prikaz članka zajedno s popisom slika koje mu pripadaju. Na stranici će biti prikazani link za izmjenu članka i link za dodavanje nove slike uz članak, a u popisu slika nalazit će se linkovi za brisanje pojedinih slika.

Na stranici *unosClanka.php* bit će prikazan obrazac preko kojeg će se moći unijeti novi članak ili izmijeniti postojeći. Obrazac će se slati istoj stranici – *unosClanka.php* koja će obaviti unos novog ili izmjenu postojećeg članka i preusmjeriti korisnika na stranicu s prikazom članka (*prikazClanka.php*).

Na stranici *unosSlike.php* nalazit će se obrazac za slanje slike na poslužitelj. Obrazac će se slati istoj stranici, koja će spremiti sliku na poslužitelj i preusmjeriti korisnika natrag na stranicu s prikazom članka.

Link za brisanje slike na stranici *prikazClanka.php* vodit će na skriptu *brisanjeSlike.php* koja će obaviti brisanje slike i vratiti korisnika natrag na stranicu s prikazom članka.

14.4. Baza podataka

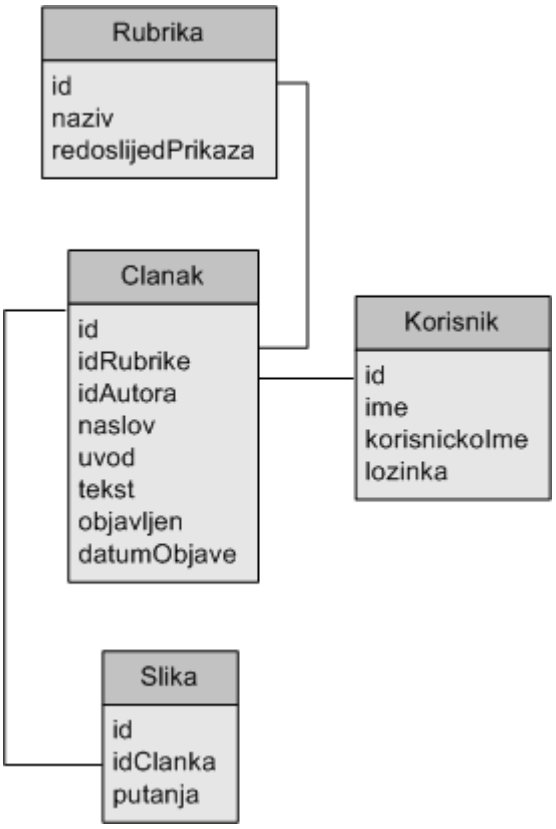
U sustavu postoje četiri entiteta: rubrika, članak, slika i korisnik.

Između rubrike i članka postoji hijerarhijska veza – svakoj rubrici pripada jedan ili više članaka. Ova je veza ostvarena na taj način da se uz članak sprema i identifikator rubrike kojoj pripada.

Slična je veza i između članka i slike – uz svaki članak može postojati jedna ili više slika. Ova je veza ostvarena tako da se uz sliku sprema i identifikator članka kojemu slika pripada.

Između korisnika i članka također postoji veza – korisnik može biti autor jednog ili više članaka. Ova je veza ostvarena na taj način da se uz članak sprema i identifikator korisnika koji je autor članka.

Dijagram baze podataka ovog sustava izgledat će ovako:



Tablica *Rubrika* sadrži ova polja:

- *id* – identifikator rubrike
- *naziv* – naziv rubrike
- *redoslijedPrikaza* – broj koji određuje po kojem redoslijedu će se rubrike ispisati u navigaciji.

Tablica *Clanak* sadrzi ova polja:

- *id* – identifikator rubrike
- *idRubrike* – identifikator rubrike kojoj članak pripada
- *idAutora* – identifikator korisnika koji je autor članka
- *naslov* – naslov članka
- *uvod* – početni dio teksta članka koji se prikazuje na stranici rubrike
- *tekst* – ostatak teksta članka
- *objavljen* – cijeli broj (0 ili 1) koji određuje je li članak vidljiv na javnom dijelu ili ne
- *datumObjave* – datum kada je članak objavljen.

Tablica *Slika* sadrži ova polja:

- *id* – identifikator slike
- *idClanka* – identifikator članka kojem slika pripada
- *putanja* – putanja do slike na poslužitelju.

Tablica *Korisnik* sadrzi ova polja:

- *id* – identifikator korisnika
- *punoIme* – puno ime korisnika
- *korisnickoIme* – ime koje se koristi za prijavu korisnika u sustav
- *lozinka* – lozinka koja se koristi za prijavu u sustav.

Vježba 14.1 – Stvaranje baze podataka za sustav za upravljanje sadržajem

1. Pokrenite servis *EasyPHP*, ako već nije pokrenut.
2. Pokrenite aplikaciju *phpMyAdmin* tako da u web preglednik upišete adresu http://localhost/mysql/.
3. Stvorite novu bazu podataka s nazivom "cms". Za *collation* baze odaberite vrijednost *utf8_general_ci*.
4. Stvorite novu tablicu u bazi podataka koja će imati naziv "Rubrika" i 3 polja. Stvorite polja prema ovom popisu:

Naziv polja	Tip podatka	Duljina	Ostalo
id	INT		auto_increment, primary key
naziv	VARCHAR	50	
redoslijedPrikaza	INT		

5. Unesite ove podatke u tablicu (klikom na "Novi redak").

naziv	redoslijed Prikaza
Lorem ipsum	0
Nullam quis felis	1
Quisque at sapien	2
Sed accumsan orci	3
Nunc luctus	4

U polje *id* nije potrebno unositi podatke jer se *id* zadaje automatski (opcija *auto_increment*).

6. Stvorite novu tablicu u bazi podataka, koja će imati naziv "Korisnik" i 4 polja. Stvorite polja prema ovom popisu:

Naziv polja	Tip podatka	Duljina	Ostalo
id	INT		auto_increment, primary key
punoIme	VARCHAR	50	
korisnickoIme	VARCHAR	50	
lozinka	VARCHAR	50	

7. U tablicu unesite jednog korisnika s ovim podacima:

punoIme	korisnickoIme	lozinka
Administrator	admin	123

8. Stvorite novu tablicu u bazi podataka koja će imati naziv "Clanak" i 8 polja. Stvorite polja prema ovom popisu:

Naziv polja	Tip podatka	Duljina	Ostalo
id	INT		auto_increment, primary key
idRubrike	INT		
idAutora	INT		
naslov	VARCHAR	100	
uvod	TEXT	300	
tekst	TEXT	2000	
objavljen	INT	1	
datumUnosa	TIMESTAMP		

9. Stvorite novu tablicu u bazi podataka koja će imati naziv "Slika" i 3 polja. Stvorite polja prema ovom popisu:

Naziv polja	Tip podatka	Duljina	Ostalo
id	INT		auto_increment, primary key
idClanka	INT		
putanja	VARCHAR	50	

Vježba 14.2 – Izrada administrativnog dijela sustava za upravljanje sadržajem

1. Pokrenite program *PHP Designer*. U njegovom pregledniku datoteka (*File Browser*) odaberite mapu "..\D350\vjezbe\vjezba14.2" . U ovoj mapi postoji 5 mapa:
   - "Admin" – u ovoj mapi će se nalaziti stranice koje čine administrativni dio sustava
   - "Frontend" – u ovoj mapi će se nalaziti stranice koje čine javno dostupni dio sustava
   - "Include" – mapa s dijelovima kôda koji se umeću u druge datoteke – pomoćne funkcije i dijelovi koji se ponavljaju na drugim stranicama
   - "Izgled" – mapa koja sadrži CSS datoteku koja definira izgled aplikacije
   - "Slike" – mapa u koju će se spremati slike poslane na poslužitelj
2. Stvorite novu datoteku ***otvoriVezu.php*** u mapi "Include". Ova datoteka sadržavat će funkciju za otvaranje veze s bazom podataka. Ova datoteka bit će uključena u svim datotekama budući da sve datoteke ili čitaju ili mijenjaju podatke iz baze. Funkcija je ista je kao i funkcija *otvoriVezu* iz vježbi u 12. poglavlju, osim što je razlika u imenu baze:

```
<?php

  function OtvoriVezu()
  {
    $veza = mysql_connect("localhost", "root", "");
    if ($veza)
    {
      if (mysql_select_db("cms", $veza))
      {
        return $veza;
      }
      else
      {
        exit("Baza s tim imenom ne postoji!");
      }
    }
    else
    {
      exit("Ne mogu se spojiti na MySql server!");
    }
  }
?>
```

3. Stvorite novu datoteku ***provjera.php*** u mapi "Include". Ova datoteka će biti uključena u svim datotekama u mapi "Admin", a služit će za provjeru je li korisnik autenticiran (na temelju toga je li mu korisničko ime zapisano u varijablu sjednice), pa ako nije, korištenjem funkcije *header* presumjerit će ga na stranicu za prijavu – *prijava.php*.

```
<?php
  session_start();

  if (!isset($_SESSION["korisnickoIme"]))
  {
    header("Location: prijava.php");
  }
?>
```

4. Stvorite novu datoteku ***zaglavlje.php*** u mapi "Include". U ovoj datoteci nalazit će se zaglavlje koje će se pojavljivati na svakoj stranici. Tu se nalazi HTML zaglavlje s oznakom kodne stranice, kao i s vezom na CSS datoteku. Zatim se ispisuje naziv aplikacije ("superCMS"). Ispod njega će se prikazivati ime prijavljenog korisnika, a ako korisnik nije prijavljen, prikazivat će se link na stranicu *prijava.php*:

```
<html>
  <head>
    <title>Sustav za upravljanje sadržajem</title>
    <meta http-equiv="Content-Type" content="text/html;
          charset=windows-1250" />
    <link rel="stylesheet" type="text/css"
     href="../Izgled/izgled.css" />
  </head>
<body>
  <h1>superCMS</h1>
  <?php
    if (isset($_SESSION["korisnickoIme"]))
    {
      echo "Prijavljen: <b>" . $_SESSION["korisnickoIme"] .
"</b>";
     }
     else
     {
       echo '<a href="../Admin/prijava.php">Prijavi se</a>';
     }
  ?>
  <hr />
```

5. Stvorite novu datoteku ***podnozje.php*** u mapi "Include", u kojoj će biti dno HTML dokumenta:

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

Datoteke *zaglavlje.php* i *podnozje.php* bit će uključene u svaku stranicu u aplikaciji koja ispisuje HTML. Između zaglavlja i podnožja nalazit će se razni HTML kôd, kao i naredbe za ispis HTML oznaka i podataka.

6. U mapi "Include" stvorite još i datoteku ***pomocneFunkcije.php*** koja će sadržavati jednu pomoćnu funkciju koja će raditi pretvaranje znakova za novi red u oznaku *br* HTML-a. Naime, tekst članka unosit će se preko višelinijskog polja za unos (*textarea*) pa će sadržavati znakove \r i \n kao oznake za novi red. Zbog toga će ga biti potrebno preformatirati pri ispisu kao HTML.

```
<?php
 function ZamijeniZnakoveZaNoviRed($ulaz)
 {
   $izlaz = str_replace("\r\n", "<br />", $ulaz);
   $izlaz = str_replace("\n", "<br />", $izlaz);

   return $izlaz;
 }
?>
```

7. Prebacite se u mapu "Admin" i stvorite datoteku ***popisClanaka.php***, koja će prikazivati sve članke u sustavu zajedno s linkovima za njihovo uređivanje. Najprije upišite izraze za uključenje potrebnih datoteka:

```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/provjera.php");
  require("../Include/zaglavlje.php");
?>
```

Važno je da se datoteka *provjera.php* uključi prije datoteke *zaglavlje.php*, zato što se u datoteci *provjera.php* (ovisno o situaciji) može pozvati funkcija header za preusmjeravanje koja će javiti grešku ako je prije nje već ispisano nešto HTML-a.

8. Nakon ovih naredbi, i poslije oznaka za PHP kôd, dolazi zaglavlje HTML tablice pomoću koje će se ispisivati postojeći članci:

```
<h2>Popis članaka</h2>
<table class="list" cellspacing="0">
  <tr>
    <th>Unešen</th>
    <th>Rubrika</th>
    <th>Naslov</th>
    <th>Objavljen</th>
    <th>Brisanje</th>
  </tr>
```

9. Nakon ovog HTML-a, ponovno dolaze oznake za PHP kôd, unutar kojih će se dohvatiti postojeći članci iz baze i ispisati redak po redak:

```
<?php
  $veza = OtvoriVezu();
  $rezultat = mysql_query("SELECT Clanak.id AS idClanka, naziv,
naslov, objavljen, UNIX_TIMESTAMP(datumUnosa) AS datum FROM
Clanak, Rubrika WHERE rubrika.id = clanak.idRubrike ORDER BY
datumUnosa DESC");
   if (mysql_num_rows($rezultat) == 0)
   {
      echo '<tr><td colspan="5" align="middle">Nema
članaka</td></tr>';
    }
    while ($redak = mysql_fetch_array($rezultat))
    {
        $id = $redak["idClanka"];
        $datum = $redak["datum"];
        $rubrika = $redak["naziv"];
        $naslov = stripSlashes($redak["naslov"]);
        if ($redak["objavljen"] == 1)
        {
          $objavljen = "Da";
        }
        else
        {
          $objavljen = "Ne";
        }
        echo "<tr>";
        echo "<td>" . date("j.n.Y.", $datum) . "</td>";
        echo "<td>$rubrika</td>";
        echo "<td><a
href='prikazClanka.php?id=$id'>$naslov</a></td>";
        echo "<td>$objavljen</td>";
        echo "<td><a href='brisanjeClanka.php?id=$id'>Obriši
članak</a></td>";
        echo "</tr>";
    }
    mysql_free_result($rezultat);
    mysql_close($veza);
?>
```

U *SELECT* upitu radi se spoj između tablica *Clanak* i *Rubrika*, preko identifikatora rubrike, i to da bi se u rezultatu upita dobio naziv rubrike kojoj članak pripada. Budući da sustav za upravljanje bazama podataka MySQL ima vlastiti vremenski format, datum unosa je potrebno pretvoriti u Unixovu vremensku oznaku (format koji koristi PHP) pomoću MySQL-ove funkcije *UNIX_TIMESTAMP*.

Ako nijedan članak ne postoji u bazi, ispisat će se poruka "Nema članaka". U suprotnom, članci će se čitati u petlji dok se svi ne ispišu u tablici. Naslov članka će biti link na stranicu s detaljnijim prikazom članka, a u posljednjem stupcu nalazit će se link za brisanje članka. Pri ispisivanju polja, naslov iz baze koristi se funkcijom *stripSlashes* zato što će se na sve podatke koji se šalju metodama *GET* ili *POST* u ostatku aplikacije primijenjivati metoda *addSlashes* prije upisa u bazu.

10. Nakon oznaka za PHP kôd potrebno je zatvoriti tablicu, a tu dolazi još i link za stvaranje novog članka i podnožje HTML dokumenta.

```
</table>
<br />
<a href="unosClanka.php">Novi članak</a>
<?php
  require("../Include/podnozje.php");
?>
```

11. U mapi "Admin" stvorite datoteku ***unosClanka.php*** koja će služiti za služiti za unos novog, ali i za izmjenu postojećeg članka. Ako u URL-u bude proslijeđen identifikator članka, skripta će morati prepoznati da se radi o izmjeni postojećeg članka. Najprije upišite izraze za uključenje potrebnih datoteka:

```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/provjera.php");
```

Zaglavlje još nećemo uključivati, jer će i ova skripta, ovisno o uvjetima, raditi preusmjeravanje na druge skripte. Ova skripta će sama obraditi podatke upisane na obrazac za unos ili izmjenu korisnika koji se nalazi na njoj i nakon spremanja podataka, preusmjeriti korisnika natrag.

Ako podaci još nisu poslani, skripta će samo prikazati obrazac.

12. Najprije se provjerava je li obrazac već ispunjen i poslan. Ako jest (i ako su ispunjeni svi podaci), možemo ih spremiti u bazu podataka:

```
if (isset($_POST["rubrika"]) && isset($_POST["naslov"]) &&
    isset($_POST["uvod"]) && isset($_POST["tekst"]))
{
  $rubrika = addSlashes($_POST["rubrika"]);
  $naslov = addSlashes($_POST["naslov"]);
  $uvod = addSlashes($_POST["uvod"]);
  $tekst = addSlashes($_POST["tekst"]);
  if (isset($_POST["objavljen"]))
  {
    $objavljen = 1;
  }
  else
  {
    $objavljen = 0;
  }

  if (isset($_POST["id"]))
  {
    // izmjena članka
    $id = addSlashes($_POST['id']);
    $sql = "UPDATE Clanak SET idRubrike = $rubrika, naslov =
'$naslov', uvod = '$uvod', tekst = '$tekst', objavljen =
$objavljen WHERE id = $id";
   }
   else
   {
     // unos novog članka
     $sql = "INSERT INTO Clanak (idRubrike, idAutora, naslov,
uvod, tekst, objavljen) VALUES ( $rubrika,
{$_SESSION['idKorisnika']}, '$naslov', '$uvod', '$tekst',
'$objavljen')";
   }

   $veza = otvoriVezu();
   if (mysql_query($sql))
   {
     header("Location: popisClanaka.php");
   }
   else
   {
     mysql_close($veza);
     exit("Unos / izmjena članka nije uspjela.");
   }
   mysql_close($veza);
  }
```

Podaci koji su bili uneseni u obrazac pomoću funkcije *addSlashes* se čine sigurnima za unos u bazu podataka. Ako je zajedno s ostalim podacima (kao skriveno polje) poslan i identifikator članka, radi se o izmjeni već postojećeg članka i obavit će se upit tipa *UPDATE*, a ako nema identifikatora, stvara se novi članak pa će se obaviti upit tipa INSERT.

Ako je upit uspio, zatvorit će se veza s bazom i obaviti preusmjeravanje na stranicu s popisom članaka. Ako upit ne uspije, ispisat će se poruka i prekinuti izvršavanje pomoću funkcije *exit*. No, i u tom slučaju je potrebno najprije zatvoriti vezu s bazom.

13. Druga mogućnost je da obrazac nije bio ispunjen i poslan, nego se tek prikazuje korisniku prvi put. U tom slučaju, ako je riječ o izmjeni postojećeg članka, treba ispisati njegove podatke unutar obrasca. Da je članak već postojeći, prepoznat će se po tome što će u URL-u biti prenesen identifikator članka:

```
  if (isset($_GET["id"]))
  {
    // prikaz podataka za postojeći članak
    $id = addSlashes($_GET['id']);
    $veza = OtvoriVezu();
    $rezultat = mysql_query("SELECT * FROM clanak WHERE id =
$id");
    if($redak = mysql_fetch_array($rezultat))
    {
      $naslovStranice = "Izmjena članka";
      $linkZaPovratak = "prikazClanka.php?id=$id";
      $idRubrike = $redak["idRubrike"];
      $naslov = stripSlashes($redak["naslov"]);
      $uvod = stripSlashes($redak["uvod"]);
      $tekst = stripSlashes($redak["tekst"]);
      if ($redak["objavljen"] == 1)
      {
        $objavljen = "checked";
      }
      else
      {
        $objavljen = "";
      }
    }
    mysql_free_result($rezultat);
    mysql_close($veza);
  }
  else if (!isset($_POST["id"]))
  {
    // novi članak
    $naslovStranice = "Unos članka";
    $linkZaPovratak = "popisClanaka.php";
    $idRubrike = 0;
    $naslov = "";
    $uvod = "";
    $tekst = "";
    $objavljen = "checked";
  }
```

Ako je predan identifikator članka, njegovi podaci se čitaju iz baze (pri čemu se miču zaštitne kose crte funkcijom *stripSlashes*) i spremaju u varijable. Ako se radi o novom članku, te će varijable poprimiti prazne vrijednosti, ili predefinirane vrijednosti (varijabla *$objavljen*).

14. Nakon što su učitane postojeće vrijednosti (ako je to bilo potrebno), možemo prikazati obrazac za unos ili izmjenu vrijednosti, i prije njega, zaglavlje HTML stranice:

```
  require("../Include/zaglavlje.php");
?>
<h2><?php echo $naslovStranice ?></h2>
```

```
<form method="POST" action="unosClanka.php">
  <?php
    if (isset($id))
    {
      echo "<input type='hidden' name='id' value='$id'>";
    }
  ?>
  <table>
    <tr>
      <td>Rubrika</td>
      <td>
        <select name="rubrika">
          <?php
            $veza = OtvoriVezu();
            $rezultat = mysql_query("SELECT * FROM rubrika
ORDER BY redoslijedPrikaza");
            while ($redak = mysql_fetch_array($rezultat))
            {
              $id = $redak["id"];
              $naziv = $redak["naziv"];

              echo '<option value="' . $id . '"';
              if ($id == $idRubrike)
              {
                echo " selected";
              }
              echo ">$naziv</option>";
            }
            mysql_free_result($rezultat);
            mysql_close($veza);
          ?>
        </select>
      </td>
    </tr>
    <tr>
      <td>Naslov</td>
      <td><input type="text" name="naslov" maxlength="100"
size="66" value="<?php echo $naslov ?>"></td>
   </tr>
   <tr>
     <td>Uvod</td>
     <td>
       <textarea name="uvod" rows="5" cols="50">
         <?php echo $uvod ?>
       </textarea>
     </td>
   </tr>
   <tr>
     <td>Tekst</td>
     <td>
       <textarea name="tekst" rows="11" cols="50">
          <?php echo $tekst ?></textarea>
     </td>
   </tr>
   <tr>
     <td>Objavljen</td>
     <td>
       <input type="checkbox" name="objavljen" <?php echo
$objavljen ?>/>
```

```
        </td>
      </tr>
      <tr>
        <td></td>
        <td>
          <input type="submit" value="Spremi"/>
          <input type="button" value="Odustani"
onclick="javascript:void(location.href = '<?php echo
$linkZaPovratak ?>')"/>
        </td>
    </tr>
  </table>
</form>

<?php
   require("../Include/podnozje.php");
?>
```

Najprije se u obrascu, ako se radi o izmjeni postojećeg članka, ispisuje skriveno polje *id* u kojem se ispisuje vrijednost identifikatora članka, zato da bi skripta koja sprema izmjene imala identifikator retka u bazi koji treba izmijeniti.

Zatim slijedi padajuća lista za odabir rubrike, koja se ispisuje tako da se pročitaju sve rubrike iz baze i da se ispiše po jedna HTML oznaka *option* za svaku rubriku. Kao atribut value oznake option, ispisuje se identifikator rubrike, a ako je rubrika odabrana, na oznaku *option* postavlja se atribut *selected*.

Zatim slijede tekstualna polja obrasca, u koja se upisuju podaci za postojeći članak, ukoliko postoje, i na kraju, kućica za označavanje koja određuje je li članak objavljen ili nije.

Dugme "Odustani" vratit će korisnika natrag ako odustane od spremanja (vratit će ga na stranicu *popisClanaka.php* ako se radi o novom članku, a na stranicu *prikazClanka.php* ako se radi o postojećem članku). Za to je bilo potrebno iskoristiti njegov atribut *onclick*, koji je postavljen na naredbu Javascripta koja će obaviti preusmjeravanje.

Na kraju se još uključuje datoteka s podnožjem HTML stranice.

15. Zatim u mapi "Admin" stvorite datoteku ***prikazClanka.php***, koja će prikazivati odabrani članak i linkove za izmjenu članka i povratak na popis članaka. Osim toga, stranica će prikazivati i popis slika koje pripadaju članku i linkove za dodavanje nove slike i brisanje postojećih. Najprije dodajte izraze za uključivanje:

```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/pomocneFunkcije.php");
  require("../Include/provjera.php");
  require("../Include/zaglavlje.php");
```

Na ovoj stranici koristit će se i pomoćna funkcija za prikaz teksta s novim redovima.

16. Dodajte provjeru je li proslijeđen identifikator odabranog članka u URL-u, i zatim pomoću njega dohvatite članak iz baze. Njegove podatke spremite u varijable:

```
   if (isset($_GET["id"]))
  {
    // prikaz podataka za članak
    $id = addSlashes($_GET['id']);
    $veza = OtvoriVezu();
    $rezultat = mysql_query("SELECT naslov, uvod, tekst,
objavljen, UNIX_TIMESTAMP(datumUnosa) AS datum FROM clanak
WHERE id = $id");
    if($redak = mysql_fetch_array($rezultat))
    {
      $naslov = stripSlashes($redak["naslov"]);
      $uvod = stripSlashes($redak["uvod"]);
      $tekst = stripSlashes($redak["tekst"]);
      if ($redak["objavljen"] == 1)
      {
        $objavljen = "Da";
      }
      else
      {
        $objavljen = "Ne";
      }
      $datum = $redak["datum"];
    }
    mysql_free_result($rezultat);
    mysql_close($veza);
  }
?>
```

17. Nakon toga, prikažite članak u HTML tablici. Ispod tablice dodajte linkove za izmjenu članka i za povratak na popis članaka:

```
<h2>Prikaz članka</h2>
<table>
  <tr>
    <td><b>Naslov</b></td>
    <td><?php echo $naslov ?></td>
  </tr>
  <tr>
    <td><b>Uvod</b></td>
    <td><?php echo ZamijeniZnakoveZaNoviRed($uvod) ?></td>
  </tr>
  <tr>
    <td><b>Tekst</b></td>
    <td><?php echo ZamijeniZnakoveZaNoviRed($tekst) ?></td>
  </tr>
  <tr>
    <td><b>Objavljen</b></td>
    <td><?php echo $objavljen ?></td>
  </tr>
  <tr>
    <td><b>Datum unosa</b></td>
    <td><?php echo date('j.n.Y. H:i', $datum) ?></td>
  </tr>
</table>
<br />
<a href="unosClanka.php?id=<?php echo $id?>">Izmjena
članka</a> |
 <a href="popisClanaka.php">Povratak</a>
```

18. Poslije prikaza članka dodajte dio koji ispisuje popis slika, te nakon toga link za dodavanje nove slike:

```
<h2>Slike uz članak</h2>
<table class="list" cellspacing="0">
  <tr>
    <th>Slika</th>
    <th>Putanja</th>
    <th>Brisanje</th>
  </tr>
<?php
  $veza = OtvoriVezu();
  $rezultat = mysql_query("SELECT * FROM Slika WHERE idClanka
= $id");
  if (mysql_num_rows($rezultat) == 0)
  {
    echo '<tr><td colspan="3" align="middle">Nema slika uz
članak</td></tr>';
  }
  while ($redak = mysql_fetch_array($rezultat))
  {
    $idSlike = $redak["id"];
    $putanja = $redak["putanja"];

    echo "<tr>";
    echo "<td><img src='$putanja' height='50px'
width='50px'/></td>";
    echo "<td>$putanja</td>";
    echo "<td><a href='brisanjeSlike.php?id=$idSlike'>Obriši
sliku</a></td>";
    echo "</tr>";
  }
  mysql_free_result($rezultat);
  mysql_close($veza);
?>
</table>
<br/>
<a href="unosSlike.php?idClanka=<?php echo $id ?>">Dodaj novu
sliku</a>
<br/><br/>
<?php
   require("../Include/podnozje.php");
?>
```

Slike se ispisuju na taj način da se iz baze dohvate sve slike koje imaju *idClanka* jednak identifikatoru članka koji je prikazan. U tablici su uz same slike, koje su prikazane smanjeno, dane i putanje do slika i linkovi za njihovo brisanje.

19. U mapi "Admin" stvorite datoteku ***unosSlike.php*** koja će prikazivati obrazac za dodavanje nove slike uz članak, ali i obavljati spremanje odabrane slike u bazu. Identifikator članka uz koji slika treba biti spremljena prosljeđuje se putem URL-a. Najprije uključite potrebne datoteke:

```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/provjera.php");
```

20. Zatim slijedi provjera je li proslijeđen identifikator odabranog članka u URL-u. Ako jest, sprema se uvarijablu *$idClanka*. Zatim se provjerava je li slika poslana putem obrasca. Ako je, podaci vezani za nju se spremaju u bazu, a slika se sprema u mapu "Slike" na poslužitelju:

```
if (isset($_GET[" idClanka "]))
{
  $idClanka = addSlashes($_GET["idClanka"]);

 if (isset($_FILES["slika"]["tmp_name"]) &&
      $_FILES["slika"]["size"] != 0)
 {
  $putanja =
           "../Slike/{$idClanka}_{$_FILES["slika"]["name"]}";

  $sql = "INSERT INTO Slika (idClanka, putanja) ";
  $sql .= "VALUES ( $idClanka, '$putanja');";

  $veza = otvoriVezu();

  if (mysql_query($sql))
  {
    if (!move_uploaded_file($_FILES["slika"]["tmp_name"],
         $putanja))
    {
       exit("Spremanje slike na poslužitelj nije uspjelo.");
    }

    header("Location: prikazClanka.php?id=" . $idClanka );
  }
  else
  {
    mysql_close($veza);
    exit("Spremanje slike u bazu nije uspjelo.");
  }
  mysql_close($veza);
 }
}
```

U naziv slike upisuje se identifikator članka. Na taj način treba paziti da samo slike spremljene uz isti članak imaju različita početna imena, dok slike vezane uz različite članke mogu imati ista.

Ako je slika uspješno spremljena, korisnika se vraća na prikaz članka s popisom slika.

21. Ako slika nije bila već poslana, prikazat će se obrazac za odabir slike i njeno slanje na poslužitelj. I ovdje se koristi dugme "Odustani" koje pomoću JavaScript naredbe vraća korisnika natrag na prikaz članka.

```
require("../Include/zaglavlje.php");
?>
<h2>Unos slike</h2>
<form method="POST"
      action="unosSlike.php?idClanka=<?php echo $idClanka ?>"
      enctype="multipart/form-data" >
  <table>
    <tr>
      <td>Slika</td>
      <td><input type="file" name="slika"/></td>
    <tr>
      <td></td>
      <td>
        <input type="submit" value="Spremi"/>
        <input type="button" value="Odustani"
              onclick="javascript:void(location.href =
             'prikazClanka.php?id=<?php echo $idClanka ?>')"/>
      </td>
    </tr>
  </table>
</form>

<?php
   require("../Include/podnozje.php");
?>
```

22. U mapi "Admin" stvorite datoteku ***brisanjeClanka.php*** koja će obrisati članak čiji joj se identifikator proslijedi u URL-u.

```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/provjera.php");

  if (isset($_GET["id"]))
  {
    $id = addSlashes($_GET["id"]);
    $veza = OtvoriVezu();

    // najprije obriši sve slike vezane za članak
    $rezultat = mysql_query("SELECT * FROM Slika WHERE
idClanka = $id");
    while ($redak = mysql_fetch_array($rezultat))
    {
        $putanja = $redak["putanja"];
        if (file_exists("../$putanja"))
        {
          unlink("../$putanja");
        }
    }
    mysql_free_result($rezultat);
    if (!mysql_query("DELETE FROM Slika WHERE idClanka =
$id"))
    {
      mysql_close($veza);
      exit("Brisanje slika nije uspjelo.");
    }

    //zatim obriši članak
    if (!mysql_query("DELETE FROM Clanak WHERE id = $id"))
    {
      mysql_close($veza);
      exit("Brisanje članka nije uspjelo.");
    }

        mysql_close($veza);
      }

  header("Location: popisClanaka.php");
?>
```

Ova skripta neće ispisati ništa (zato ne uključuje *zaglavlje.php* i *podnozje.php*), već će samo obrisati zadani članak zajedno s pripadajućim slikama. Da bi se obrisala slika, potrebno je znati njenu putanju pa se zbog toga najprije iz baze podataka dohvate sve slike koje je potrebno obrisati. Slike se brišu s poslužitelja pomoću funkcije *unlink*. Kada se pobrišu sve slike, briše se i članak, a potom se korisnika preusmjerava na popis članaka.

23. U mapi "Admin" stvorite datoteku ***brisanjeSlike.php*** koja će obrisati sliku čiji joj se identifikator proslijedi u URL-u.

```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/provjera.php");

  if (isset($_GET["id"]))
  {
    $id = addSlashes($_GET["id"]);

    $veza = OtvoriVezu();
    $rezultat = mysql_query("SELECT idClanka, putanja FROM
slika WHERE id = $id");
    if($redak = mysql_fetch_array($rezultat))
    {
      $idClanka = $redak["idClanka"];
      $putanja = $redak["putanja"];
    }
    else
    {
      mysql_close($veza);
      exit("Slika nije pronađena u bazi.");
    }
    mysql_free_result($rezultat);

    if (mysql_query("DELETE FROM Slika WHERE id = $id"))
    {
      if (file_exists($putanja))
      {
        unlink($putanja);
      }
    }
    else
    {
      mysql_close($veza);
      exit("Brisanje slike nije uspjelo.");
     }

     mysql_close($veza);

   header("Location: prikazClanka.php?id=" . $idClanka );
  }
?>
```

Najprije je potrebno dohvatiti iz baze podataka podatke za odabranu sliku. Na temelju njene putanje uklanja je se s poslužitelja, a potom se briše i iz baze. Nakon toga korisnika se vraća na prikaz članka uz koji je ta slika bila povezana.

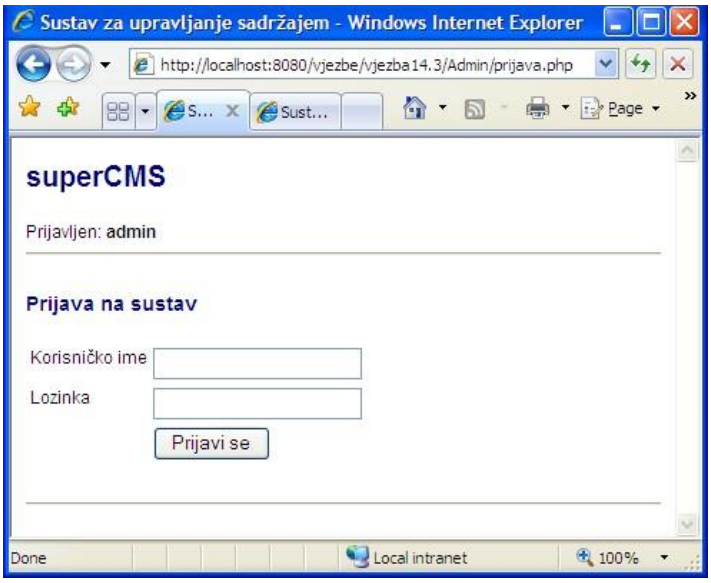
24. U mapi "Admin" stvorite novu datoteku *prijava.php*.

```
<?php
  require("../Include/otvoriVezu.php");
  session_start();
  $poruka = "";
  if (isset($_POST["korisnickoIme"]) &&
      isset($_POST["lozinka"]))
  {
    $korisnickoIme = addSlashes($_POST["korisnickoIme"]);
    $lozinka = addSlashes($_POST["lozinka"]);
    $veza = OtvoriVezu();
    $rezultat = mysql_query("SELECT * FROM korisnik WHERE
korisnickoIme = '$korisnickoIme' AND lozinka = '$lozinka'");
    if(mysql_num_rows($rezultat) == 1)
    {
      $korisnik = mysql_fetch_array($rezultat);
      $_SESSION["korisnickoIme"] = $korisnickoIme;
      $_SESSION["idKorisnika"] = $korisnik["id"];
      header("Location: popisClanaka.php");
    }
    else
    {
      $poruka = "Neuspješna prijava";
    }
    mysql_free_result($rezultat);
    mysql_close($veza);
  }
  require("../Include/zaglavlje.php");
?>
<h2>Prijava na sustav</h2>
<form method="POST" action="prijava.php">
<table>
   <tr>
      <td>Korisničko ime</td>
      <td><input type="text" name="korisnickoIme"></td>
   </tr>
   <tr>
      <td>Lozinka</td>
      <td><input type="password" name="lozinka"/></td>
   </tr>
   <tr>
      <td></td>
      <td><input type="submit" value="Prijavi se"/></td>
   </tr>
</table>
<?php echo $poruka ?>
</form>
<?php
    require("../Include/podnozje.php");
?>
```

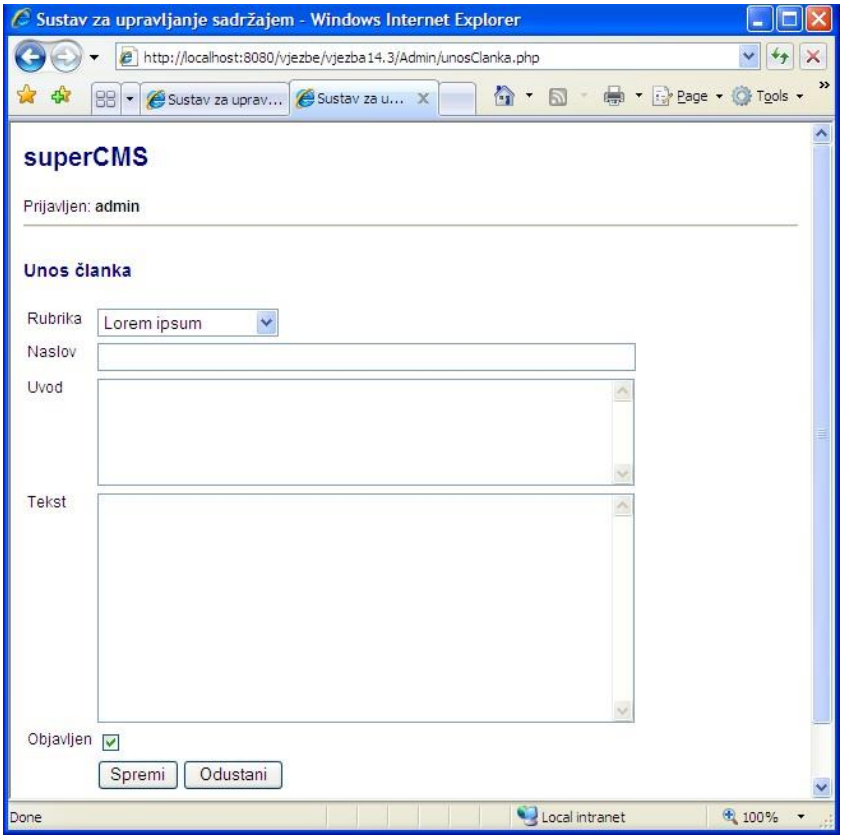
Ako su uneseni, korisničko ime i lozinka se provjeravaju u bazi. Ako su korisnički podaci pronađeni, podaci se spremaju u varijable sjednice da bi se znalo da je korisnik autenticiran. Nakon toga, korisnika se preusmjerava na stranicu *popisClanaka.php*.

25. Pokrenite servis *EasyPHP*, ako već nije pokrenut

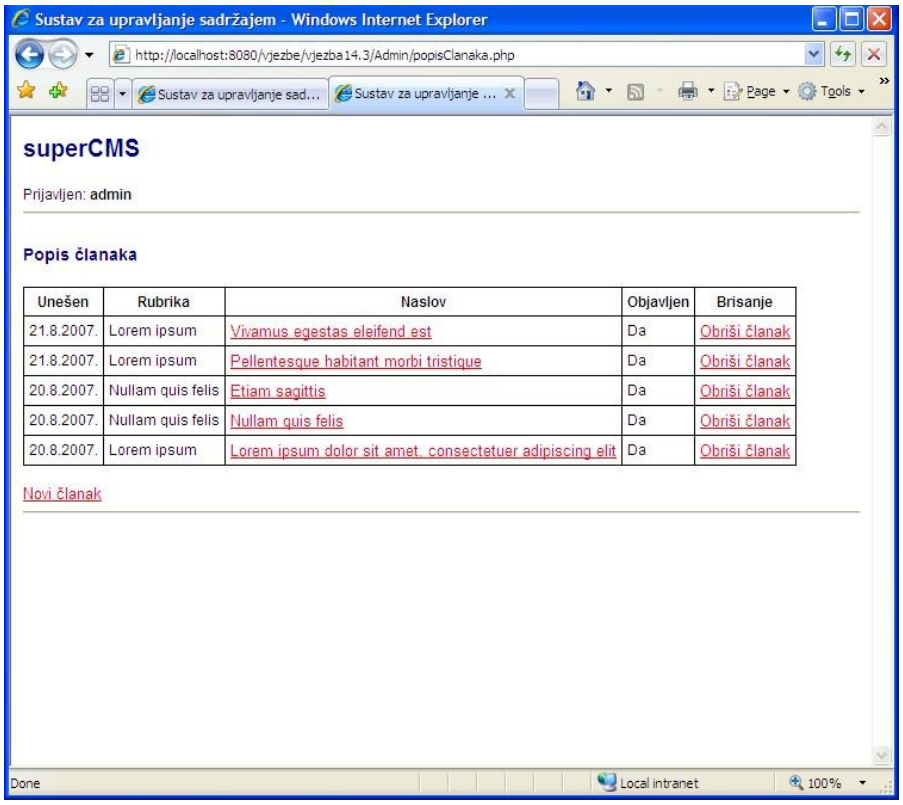
26. U web preglednik upišite adresu:
http://localhost/vjezbe/vjezba14.2/Admin/UnosClanka.php .



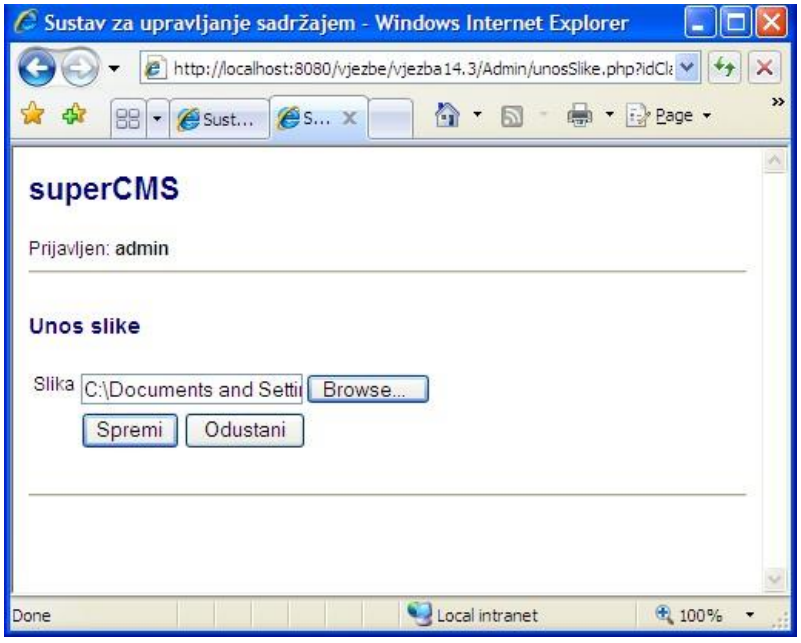
27. Prijavite se s korisničkim imenom "admin" i lozinkom "123".



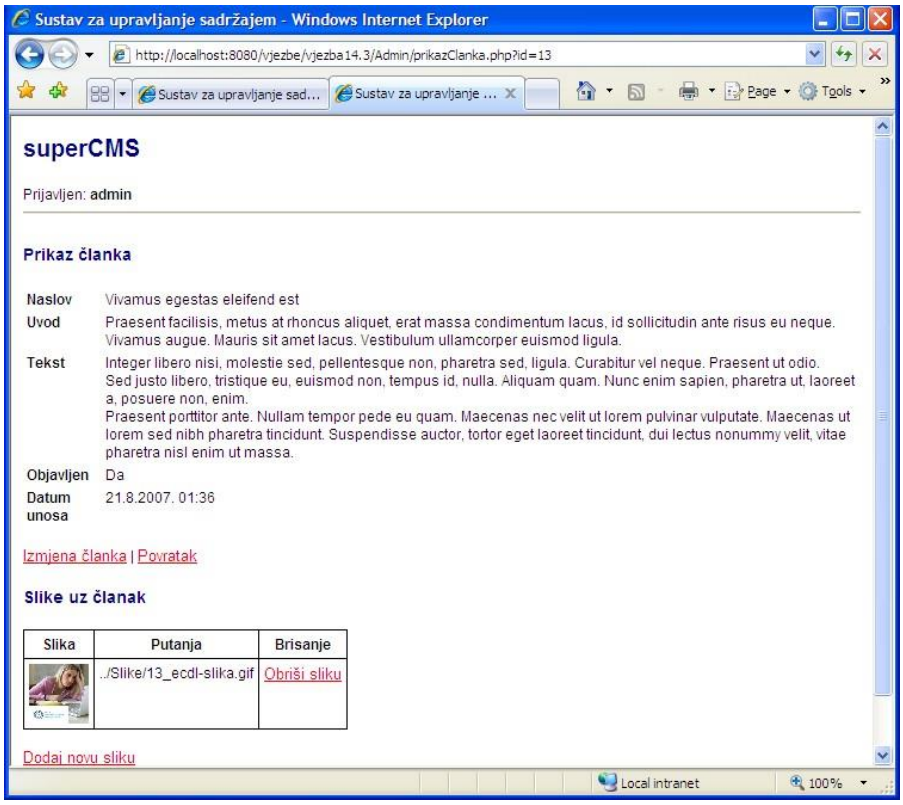
28. Unesite nekoliko novih članaka. U mapi "..\D350\vjezbe\slike\cms" nalazi se datoteka s latinskim tekstom ("Lorem ipsum") koji možete koristiti za popunjavanje članaka, kao i slike koje možete dodati uz članke.

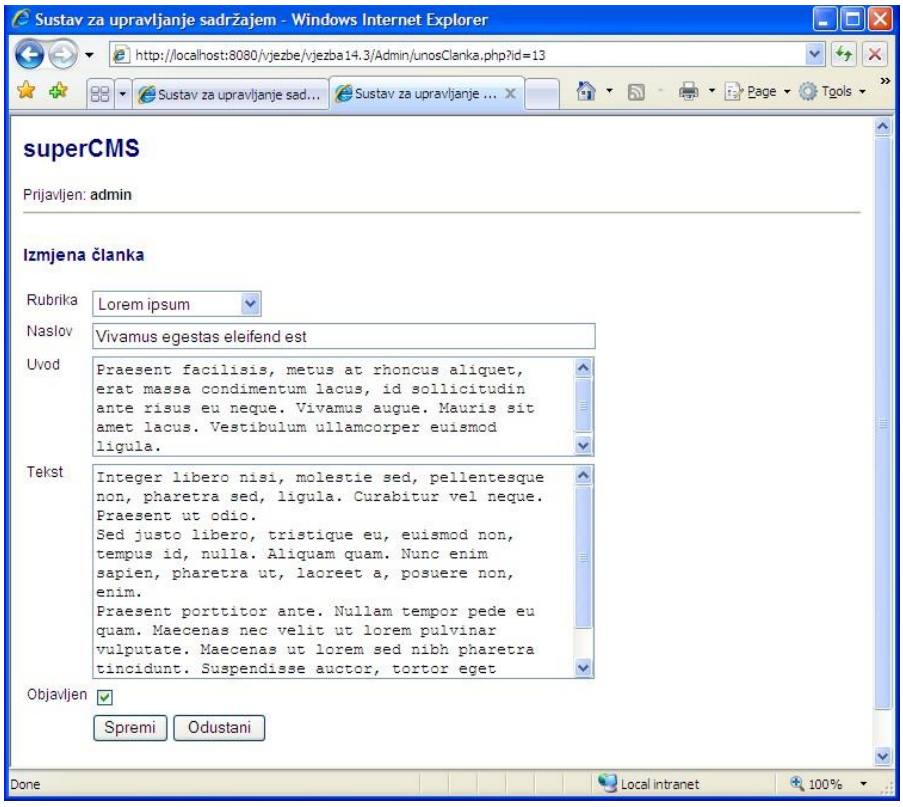


29. Uz članke dodajte i slike.



30. Isprobajte i mijenjanje članaka te brisanje članaka i slika.





Vježba 14.3 – Izrada javnog dijela sustava za upravljanje sadržajem

1. U pregledniku datoteka programa *PHP Designer* odaberite mapu "..\D350\vjezbe\vjezba14.3" .

2. U mapi "Include" stvorite datoteku ***navigacija.php*** koja će služiti za prikaz linkova na pojedine rubrike:

```
 <td id="navigacija">
  <a href="index.php">Početna</a>
  <?php
    $veza = OtvoriVezu();
    $rezultat = mysql_query("SELECT * FROM Rubrika ORDER BY
redoslijedPrikaza");

    while ($redak = mysql_fetch_array($rezultat))
    {
      $idRubrike = $redak["id"];
      $naziv = $redak["naziv"];

      echo "<a href='index.php?id=$idRubrike'>$naziv</a>";
    }
    mysql_free_result($rezultat);
    mysql_close($veza);
  ?>
</td>
```

Prvi link koji se prikazuje ne sadrži identifikator rubrike, već je to link za početnu stranicu, *index.php*. Zatim se u datoteci otvara veza s bazom i čitaju se sve postojeće rubrike (sortirane po polju *redoslijedPrikaza*). Za svaku rubriku prikazuje se link na stranicu *index.php* koji u URL-u šalje identifikator rubrike.

3. U mapi "FrontEnd" stvorite datoteku ***index.php*** koja će služiti za prikazivanje početne stranice i pojedinih rubrika. Uključite potrebne datoteke: zaglavlje, pomoćne funkcije i otvaranje veze na bazu. Datoteku s navigacijom umetnite unutar tablice:

```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/pomocneFunkcije.php");
  require("../Include/zaglavlje.php");
?>
<table>
  <tr>
    <?php
      require("../Include/navigacija.php");
    ?>
    <td id="glavniDio">
```

4. Zatim dodajte *if* strukturu koja odlučuje koji će se SQL upit koristiti, ovisno o tome radi li se o početnoj stranici ili o nekoj rubrici:

```
<?php
  if (isset($_GET["id"]))
  {
    $id = addSlashes($_GET["id"]);
    $sql = "SELECT Clanak.id AS idClanka, naslov, uvod, tekst,
UNIX_TIMESTAMP(datumUnosa) AS datum, punoIme FROM Clanak,
Korisnik WHERE korisnik.id = idAutora AND idRubrike = $id AND
objavljen = 1 ORDER BY datumUnosa DESC";
  }
  else
  {
     $sql = "SELECT  Clanak.id AS idClanka, naslov, uvod,
tekst, UNIX_TIMESTAMP(datumUnosa) AS datum, punoIme FROM
Clanak, Korisnik WHERE korisnik.id = idAutora AND objavljen =
1 ORDER BY datumUnosa DESC LIMIT 5";
  }
```

Prvi upit vraća sve članke unutar pojedine rubrike dok drugi upit (za početnu stranicu) vraća posljednjih 5 članaka iz bilo koje rubrike. U oba slučaja za članke mora vrijediti uvjet da je polje *objavljen* jednako 1.

Ovi upiti spajaju tablice *Clanak* i *Korisnik* na temelju identifikatora autora članka da bi se uz članak moglo ispisati ime autora.

5. Nakon što je upit složen, izvršava se i njegovi rezultati se ispisuju:

```
  $veza = otvoriVezu();
  $rezultat = mysql_query($sql);
  while ($redak = mysql_fetch_array($rezultat))
  {
    $idClanka = $redak["idClanka"];
    $naslov = stripSlashes($redak["naslov"]);
    $uvod = stripSlashes($redak["uvod"]);
    $tekst = stripSlashes($redak["tekst"]);
    $datum = $redak["datum"];
    $autor = stripSlashes($redak["punoIme"]);

    echo "<h2>$naslov</h2>";
    echo "<b>Objavljen</b>: " . date("j.n.Y. H:i", $datum);
    echo "<br /><b>Autor</b>: $autor";

    $slike = mysql_query("SELECT * FROM Slika WHERE idClanka =
$idClanka");
    if ($slika = mysql_fetch_array($slike))
    {
      $putanja = $slika["putanja"];
      echo "<img src='$putanja' width='100px'
height='100px'/>";
    }
    echo "<p>$uvod</p>";
    echo "<a href='clanak.php?id=$idClanka'>Više...</a>";
    echo "<div class='cleaner'></div>";
  }
  mysql_free_result($rezultat);
  mysql_close($veza);
?>
</td>
</tr>
</table>
<?php
  require("../Include/podnozje.php");
?>
```

Uz svaki članak ispisuje se njegov naslov, datum unosa, autor te uvodni dio članka. Uz članak se ispisuje i jedna slika, ako postoji, a link "Više..." vodi na stranicu gdje je članak ispisan u cijelosti sa svim slikama.

6. Pokrenite servis *EasyPHP*, ako već nije pokrenut

7. U web preglednik upišite adresu:
http://localhost/vjezbe/vjezba14.3/FrontEnd/ .

8. Posjetite nekoliko različitih rubrika. Primjećujete da se na svakoj pojavljuju različiti članci.

9. U mapi "FrontEnd" stvorite datoteku ***clanak.php*** koja će služiti za prikazivanje pojedinog članka. Uključite potrebne datoteke:

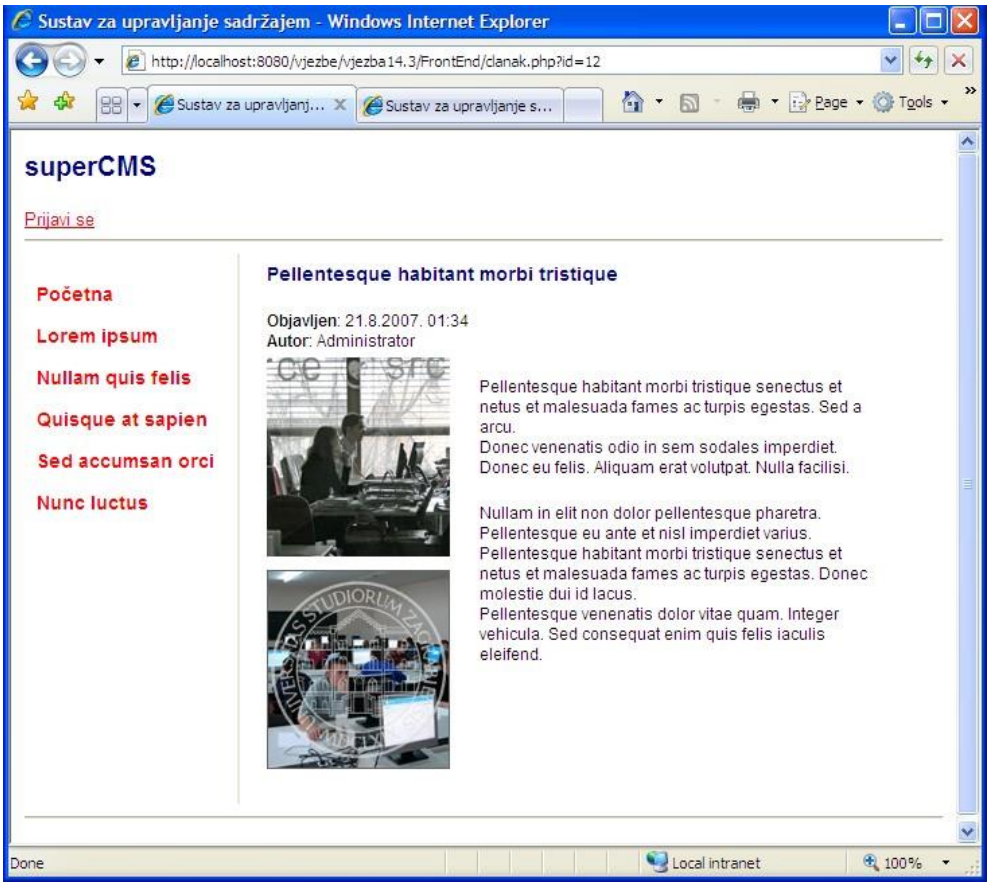
```
<?php
  require("../Include/otvoriVezu.php");
  require("../Include/pomocneFunkcije.php");
  require("../Include/zaglavlje.php");
?>
<table>
  <tr>
    <?php
      require("../Include/navigacija.php");
    ?>
    <td id="glavniDio">
```

10. Zatim se provjerava je li u URL-u proslijeđen identifikator članka. Ako jest, dohvaća se članak iz baze (tablica *Clanak* spaja se s tablicom *Korisnik* kao i kod prošlog upita). Prikazuje se puni članak zajedno s naslovom, uvodom, tekstom, autorom i vremenom objave te slikama koje će se nalaziti u lijevom dijelu prostora za članak. Dohvaćaju se sve slike i čitaju se u petlji.

```
<?php
  if (isset($_GET["id"]))
  {
    $id = addSlashes($_GET["id"]);
    $sql = "SELECT Clanak.id AS idClanka, naslov, uvod, tekst,
UNIX_TIMESTAMP(datumUnosa) AS datum, punoIme FROM Clanak,
Korisnik WHERE Korisnik.id = idAutora AND Clanak.id = $id";

  $veza = otvoriVezu();
  $rezultat = mysql_query($sql);
  if ($redak = mysql_fetch_array($rezultat))
  {
    $idClanka = $redak["idClanka"];
    $naslov = stripSlashes($redak["naslov"]);
    $uvod = stripSlashes($redak["uvod"]);
    $tekst = stripSlashes($redak["tekst"]);
    $datum = $redak["datum"];
    $autor = stripSlashes($redak["punoIme"]);

    echo "<h2>$naslov</h2>";
    echo "<b>Objavljen</b>: " . date("j.n.Y. H:i", $datum);
    echo "<br /><b>Autor</b>: $autor";

    echo "<div style='float:left'>";
    $slike = mysql_query("SELECT * FROM Slika WHERE idClanka =
$idClanka");
    while ($slika = mysql_fetch_array($slike))
    {
      $putanja = $slika["putanja"];
      echo "<img src='$putanja'/>";
    }
    echo "</div>";
    echo "<p>" . ZamijeniZnakoveZaNoviRed($uvod) . "</p>";
    echo "<p>" . ZamijeniZnakoveZaNoviRed($tekst) . "</p>";
    echo "<div class='cleaner'></div>";
  }
      mysql_free_result($rezultat);
      mysql_close($veza);
      }
?>
</td>
</tr>
</table>
<?php
  require("../Include/podnozje.php");
?>
```

Na kraju se zatvara tablica i ispisuje podnožje.

11. Kliknite na link "Više" na nekoliko članaka i pregledajte izgled stranice koja prikazuje članak.



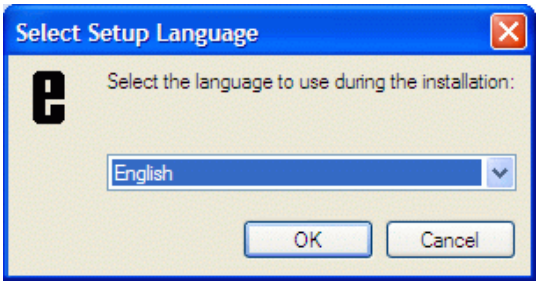
Dodatak: Upute za instalaciju programa *Easy PHP* i *PHP Designer 2007 Personal*

Instalacija programskog paketa *Easy PHP*

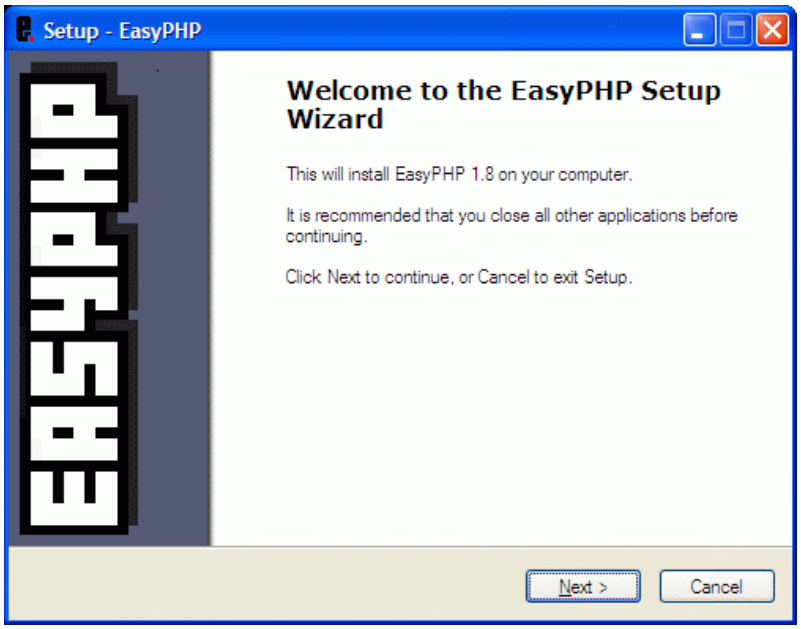
Easy PHP je besplatni programski paket koji sadrži *web* poslužitelj Apache, interpreter za PHP i sustav za upravljanje bazama podataka MySQL.

Da biste instalirali EasyPHP, slijedite ove korake (ukoliko imate CD s programima dobiven na tečaju, preskočite prvi korak):

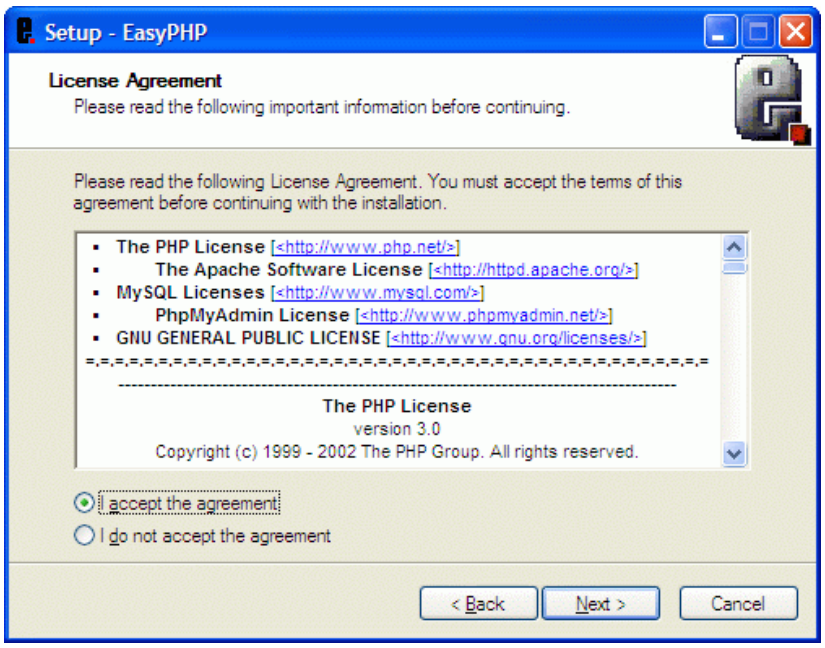
1. Pri vrhu stranice http://www.easyphp.org/telechargements.php3 pronađite link za preuzimanje posljednje inačice paketa EasyPHP (koja nije beta verzija). U ovom tečaju koristi se inačica 1.8. (lipanj 2007.)
2. Pokrenite datoteku easyphp1-8_setup.exe. (Nalazi se u mapi EasyPHP na CD-u dobivenom na tečaju.)
3. Odaberite jezik instalacije i kliknite na "OK".



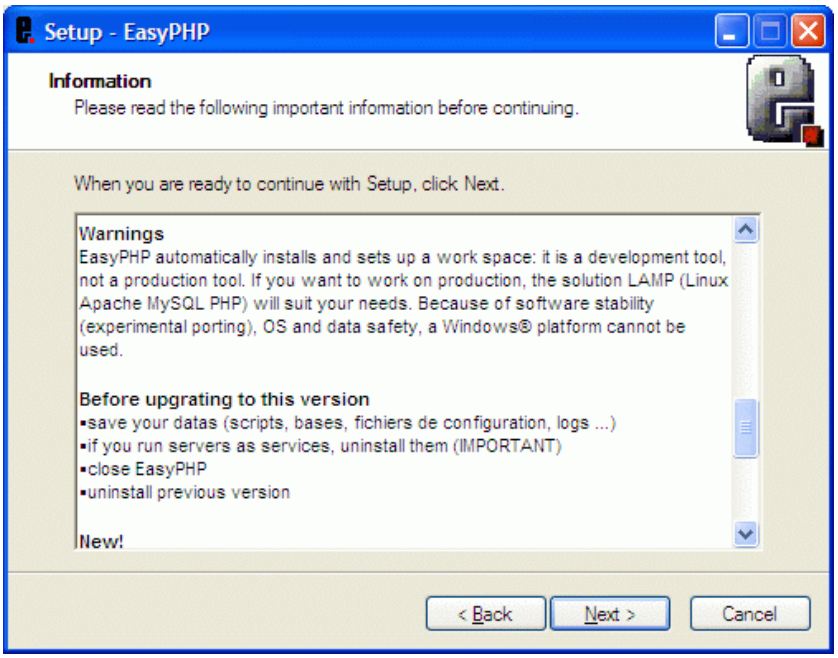
4. Kliknite "Next".



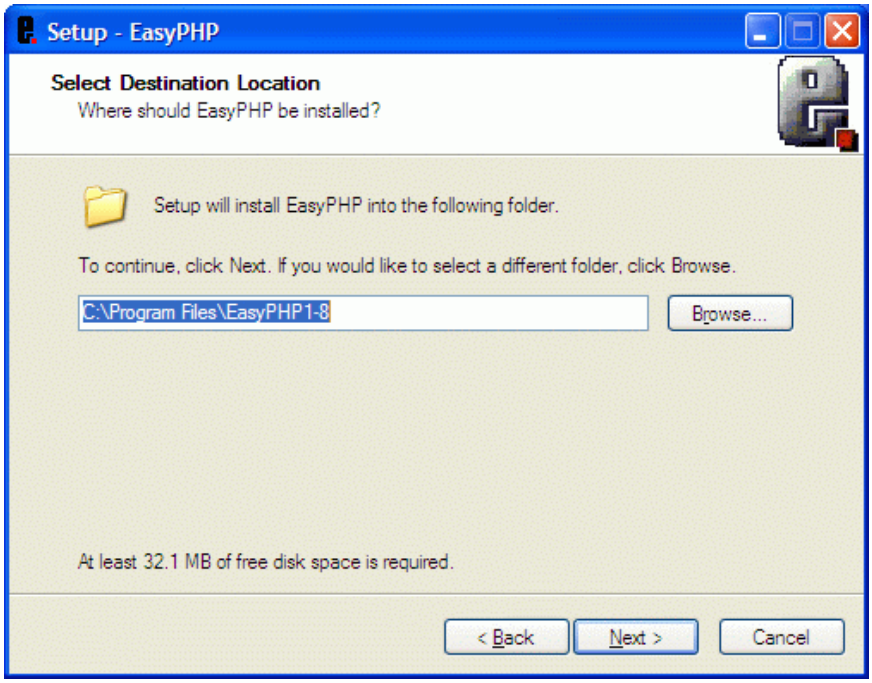
5. Odaberite "I accept the agreement" i potom kliknite "Next".



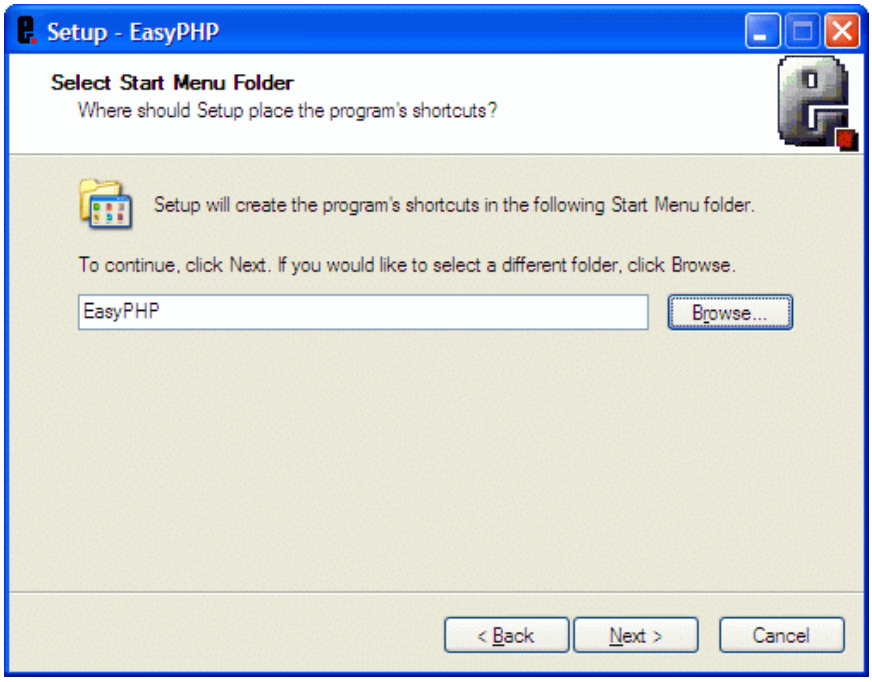
6. Kliknite "Next".



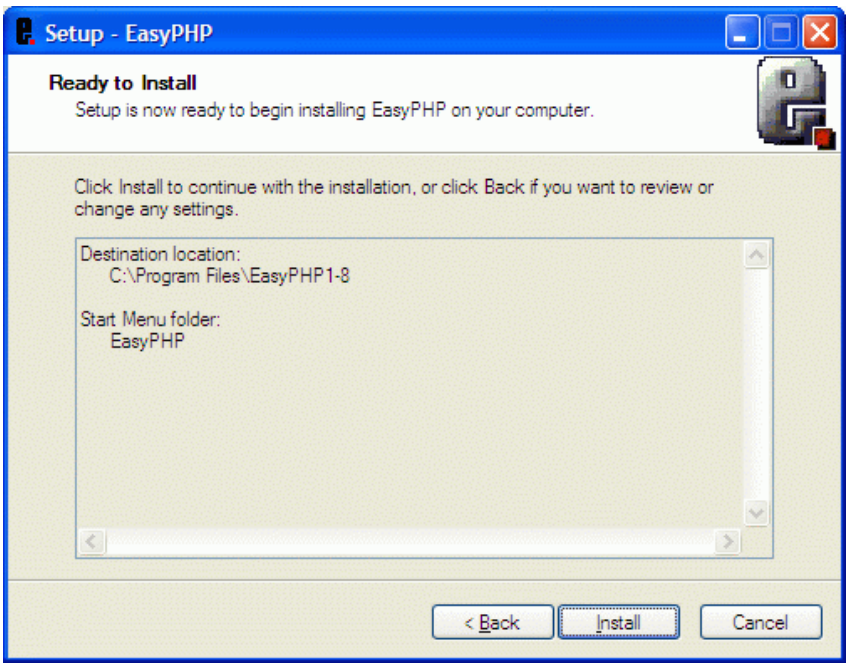
7. Odaberite mjesto na koje želite instalirati EasyPHP (npr. u mapu C:\Program Files\) i kliknite "Next".



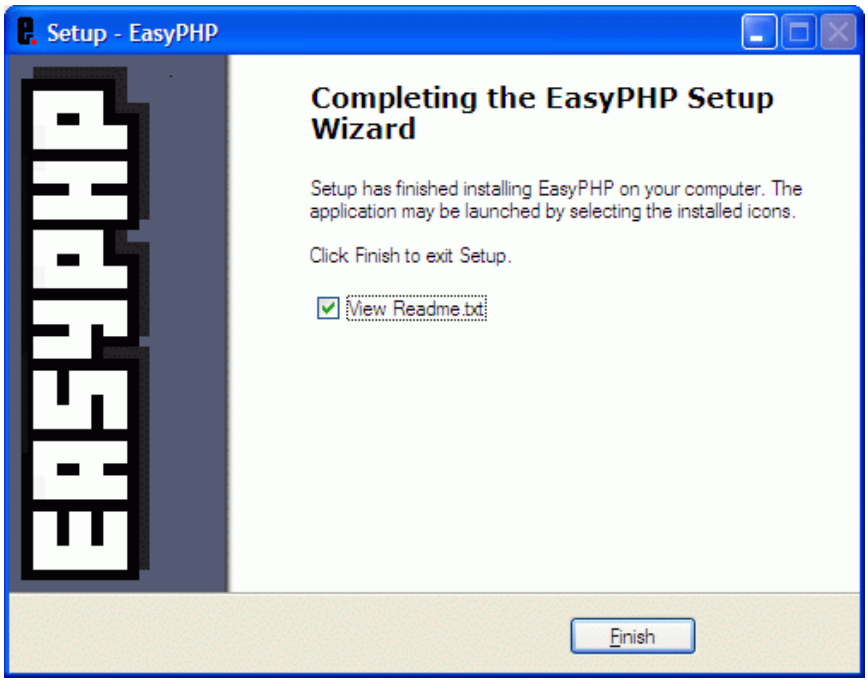
8. Odaberite mapu u izborniku Start u kojoj želite stvoriti prečac za pokretanje programa i kliknite "Next".



9. Kliknite "Install" i pričekajte da instalacija bude obavljena.

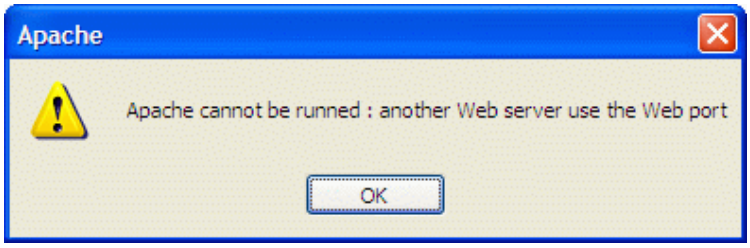


10. Ukoliko je postupak instalacije bio uspješan, pojavit će se ovakav prozor. Nakon toga kliknite „Finish“.



Mogući problemi prilikom instalacije programskog paketa EasyPHP

Ako na vašem računalu već imate instaliran neki web poslužitelj koji koristi port 80 (uobičajeni port za web promet), umjesto prozora za završetak instalacije (na prethodnoj slici) dobit ćete ovakvu poruku:



U tom slučaju kliknite na "OK" i otvorite konfiguracijsku datoteku *httpd.conf*. (Ako ste EasyPHP instalirali u mapu "C:\Program Files\EasyPHP1-8", ova će se datoteka nalaziti u mapi "C:\Program Files\EasyPHP1-8\conf_files"). Možete je otvoriti pomoću programa *Blok za pisanje* (*Notepad*). Pronađite sljedeći redak u datoteci *httpd.conf*:

```
Port 80
```

Promijenite ga tako da odaberete drugi port preko kojeg će se pristupati web serveru, npr. port 8080:

```
Port 8080
```

Pronađite i ovaj redak:

```
Listen 127.0.0.1:80
```

Gornji redak označava da web server očekuje zahtjeve na mrežnoj adresi 127.0.0.1 (adresa lokalnog računala), na portu 80. Promijenite ga u:

```
Listen 127.0.0.1:8080
```

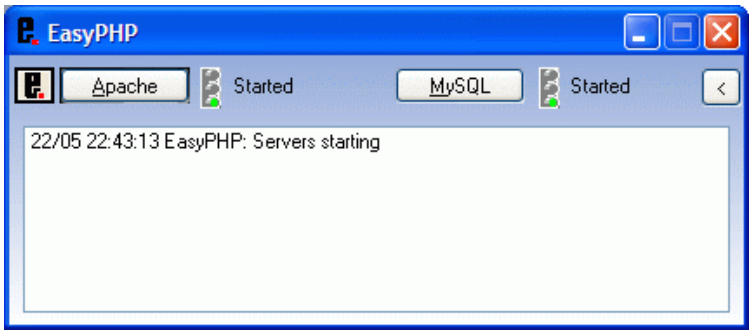
Snimite i zatvorite datoteku *httpd.conf*.

U slučaju da se ponovno javi ista pogreška (ako je i port 8080 iz nekog razloga zauzet), pokušajte upotrijebiti drugi port – bilo koji broj između 1024 i 49152 trebao bi biti u redu. Ponovno pokretanje (restart) računala nakon instalacije također može pomoći.

Pokretanje programskog paketa EasyPHP

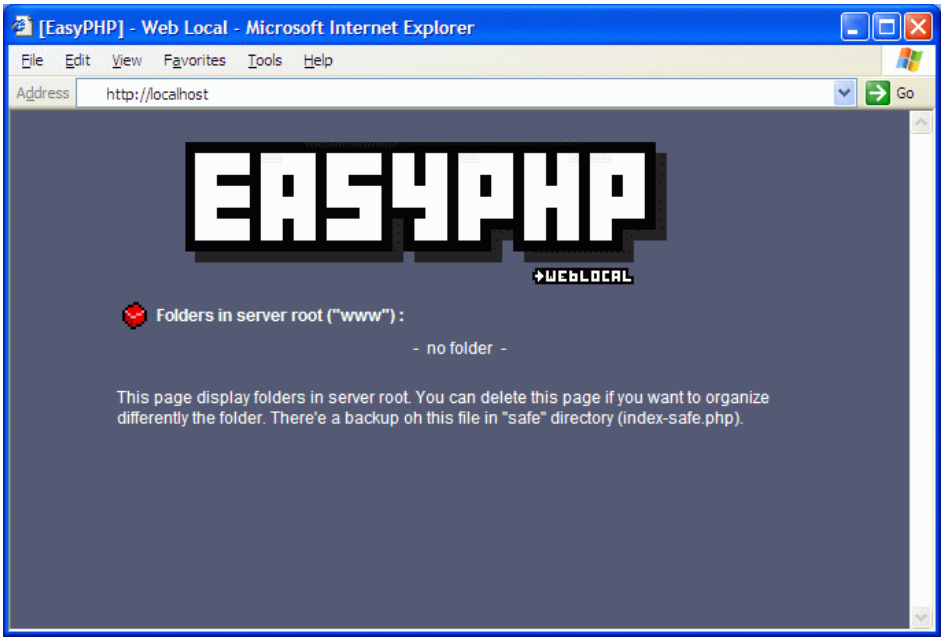
Pokrenite EasyPHP preko izbornika Start (Start > All programs > EasyPHP > EasyPHP).

Nakon uspješnog pokretanja, vidjet ćete ovaj prozor:



Zelena svjetla na oba semafora označavaju da su i web poslužitelj (Apache) i poslužitelj baza podataka (MySQL) uspješno pokrenuti.

U adresnu traku internetskog preglednika upišite "http://localhost". Ako ste morali podesiti drugi port od predodređenog, u adresnu traku upišite "http://localhost:8080", odnosno "http://localhost:*brojporta*". Ako je sve u redu, prikazat će vam se stranica s logotipom programa EasyPHP.



U vašem pregledniku je pokrenuta datoteka *index.php* koja se nalazi u korijenskoj mapi web poslužitelja ("C:\Program Files\EasyPHP1-8\www"). Unutar te mape potrebno je spremati sve PHP datoteke i druge datoteke koje se pokreću preko web poslužitelja (moguće ih je spremati i unutar podmapa).

Index.php je naziv predodređenog dokumenta (slično kao index.htm ili index.html) koji se pokreće ako se navede samo putanja do mape, bez naziva traženog dokumenta.

Nakon pokretanja servisa EasyPHP-a njegov prozor se ne smije zatvoriti (jer se time zaustavlja program), ali se može minimizirati klikom na dugme za minimiziranje prozora.

Promjena korijenske mape web poslužitelja

Korijenska mapa web poslužitelja je mapa na koju pokazuje adresa http://localhost (odnosno http://localhost:*brojporta*, ako je postavljen drugi broj porta u datoteci *httpd.conf*). Inicijalno je to mapa *www* u mapi gdje je instaliran programski paket EasyPHP. Njena putanja će biti "C:\Program Files\EasyPHP1-8\www", ako je EasyPHP instaliran u mapu "C:\Program Files\EasyPHP1-8\". Ovu putanju može te promijeniti na sljedeći način:

1. Otvorite konfiguracijsku datoteku *httpd.conf*. (u mapi "C:\Program Files\EasyPHP1-8\conf_files"). Možete je otvoriti pomoću programa *Blok za pisanje*. Pronađite sljedeći redak u datoteci *httpd.conf*:

   ```
   DocumentRoot "${path}/www"
   ```

2. Promijenite ga tako da upišete putanju do željene mape:

   ```
   DocumentRoot "D:/PHPTecaj/Vjezbe/www"
   ```

3. Snimite i zatvorite datoteku. Adresa http://localhost sada će pokazivati na odabranu mapu.

Podešavanje kôdne stranice

Kako bi podaci koji se razmjenjuju između poslužitelja i klijenta bili u ispravnom obliku (s naglaskom na hrvatske dijakritičke znakove), potrebno je u postavkama poslužitelja odabrati kôdnu stranicu (način zapisivanja znakova u binarnom obliku). Odabrana kôdna stranicu bit će navedena u zaglavlju HTTP zahtjeva.

Odabranu kôdnu stranicu možete postaviti na sljedeći način:

1. Otvorite konfiguracijsku datoteku *php.ini*. (u mapi "C:\Program Files\EasyPHP1-8\conf_files"). Možete je otvoriti pomoću programa *Blok za pisanje*. Pronađite sljedeći redak u datoteci *httpd.conf*:

   ```
   ;default_charset = "iso-8859-1"
   ```

2. Promijenite kôdnu stranicu u *windows-1250*:

   ```
   ;default_charset = "windows-1250"
   ```

3. Maknite točku-zarez s početka retka (kako bi postavka bila važeća):

   ```
   default_charset = "windows-1250"
   ```

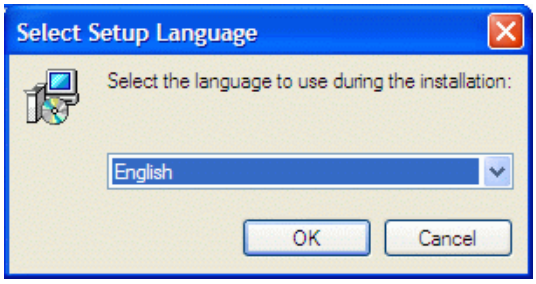
4. Snimite i zatvorite datoteku.

Instalacija programa PHP Designer 2007 Personal

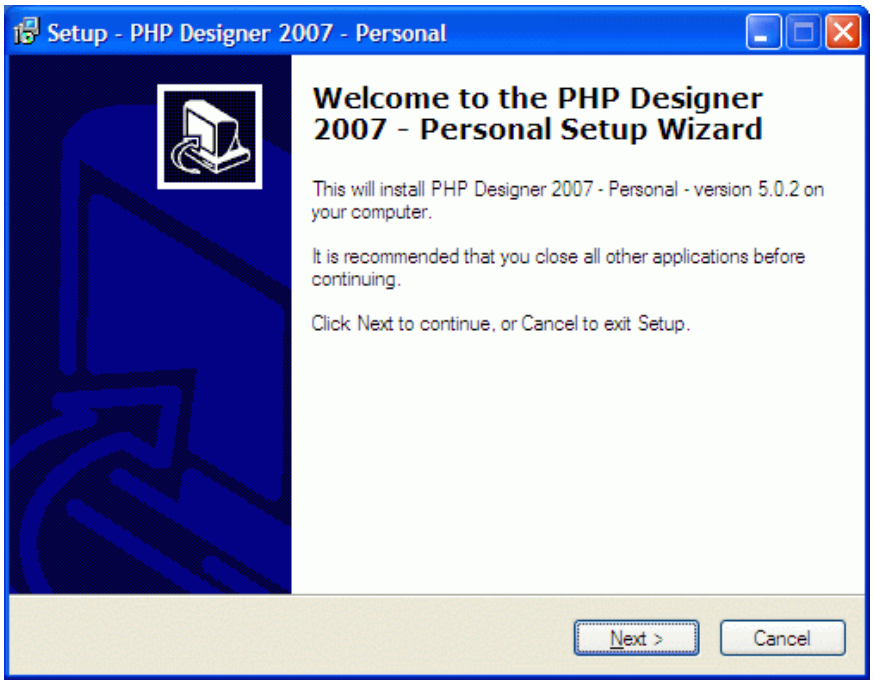
PHP Designer 2007 Personal je besplatni uređivač PHP datoteka (može poslužiti i za uređivanje brojnih drugih tipova datoteka).

Ako imate CD s instalacijskim programima koji ste dobili na tečaju, preskočite korak 1.

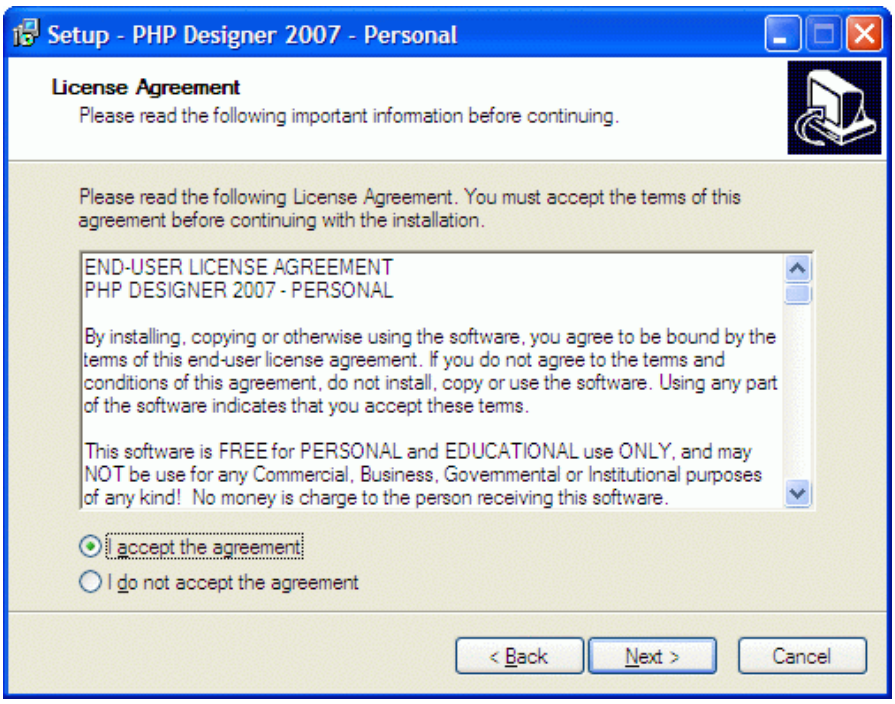
1. Na stranici http://www.mpsoftware.dk/phpdesigner_personal.php odaberite link "Free Download" i preuzmite instalacijsku datoteku.
2. Pokrenite instalacijsku datoteku *phpdesigner2007pe_setup.exe*. (Na CD-u dobivenom na tečaju nalazi se u mapi PHP Designer.)
3. Odaberite jezik instalacije i kliknite na "OK".



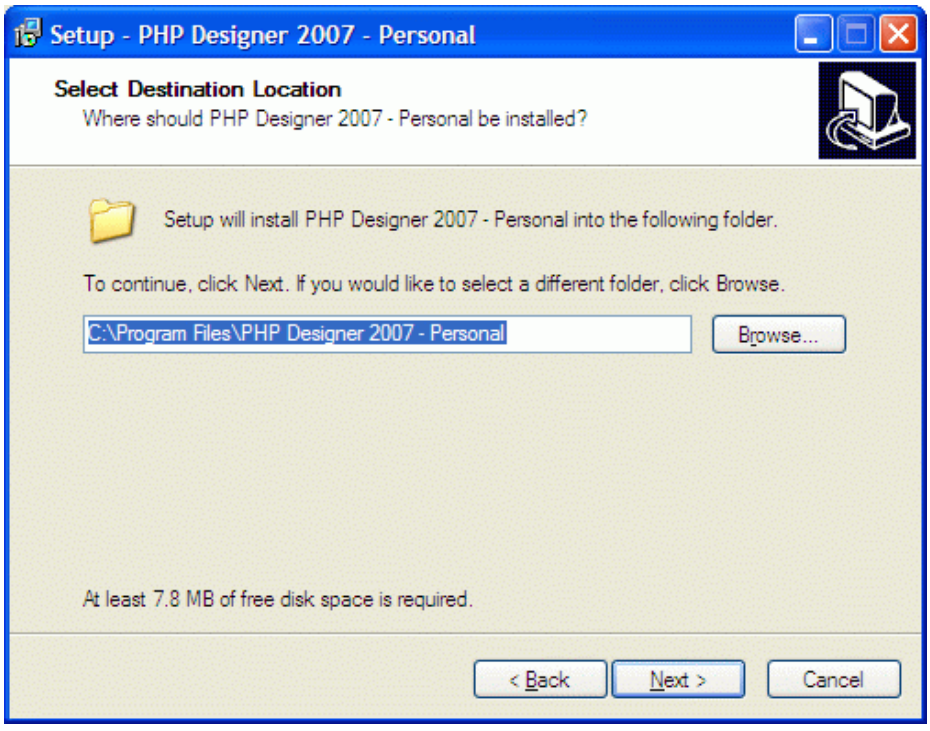
4. Kliknite "Next".



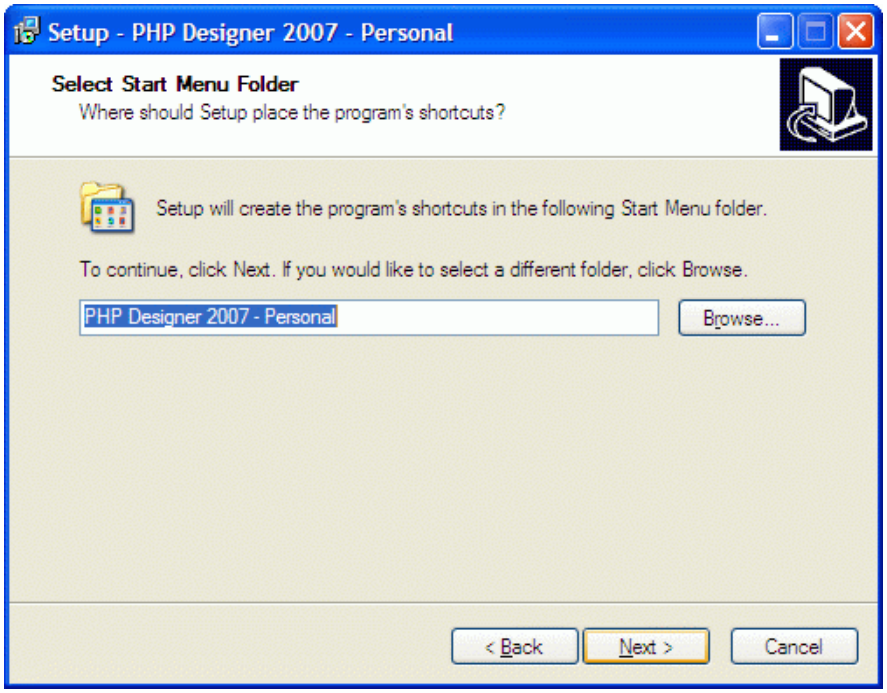
5. Odaberite "I accept the agreement" i potom kliknite "Next".



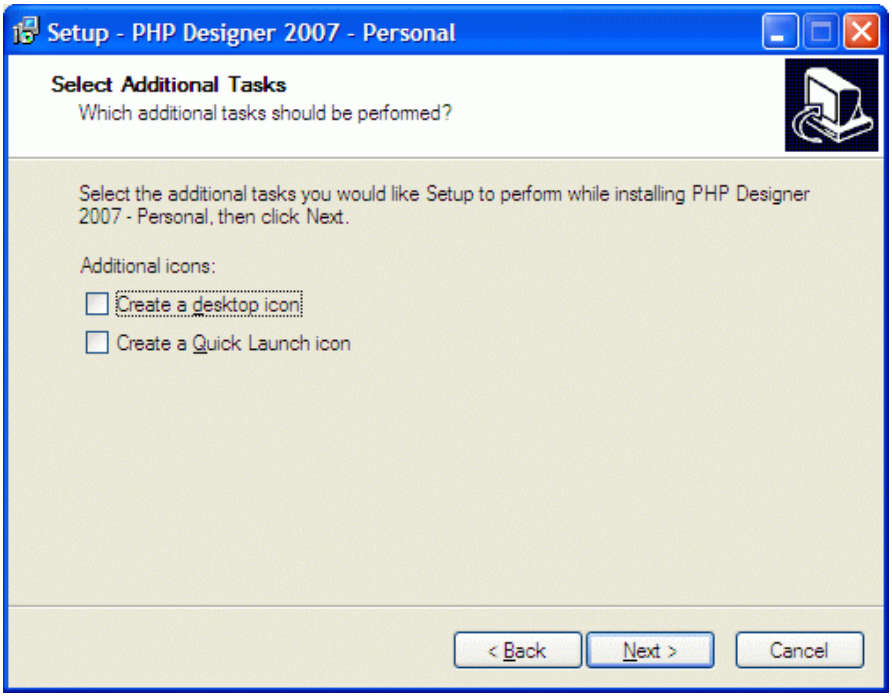
6. Odaberite mjesto na koje želite instalirati PHP Designer (npr. u mapu C:\Program Files\) i kliknite "Next".



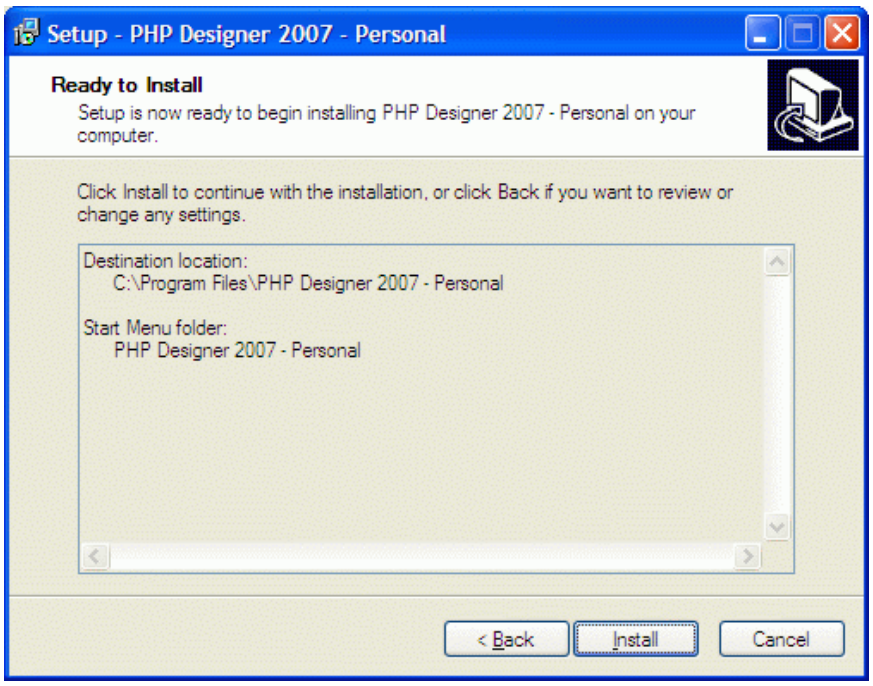
7. Odaberite mapu u izborniku Start u kojoj želite stvoriti prečac za pokretanje programa i kliknite "Next".



8. Odaberite želite li stvoriti prečac na radnoj površini te prečac za brzo pokretanje, i kliknite "Next".



9. Kliknite na "Install" i pričekajte da se instalacija završi.



10. Pokrenite PHP Designer klikom na "Finish".