Uvod u PHP i MySQL

D351

OBRAZOVNI PROGRAMI srca

priručnik za polaznike

srce
Sveučilište u Zagrebu
Sveučilišni računski centar

Ovu su verziju priručnika izradili:
Autor: Krešimir Tkalec
Recenzentica: Lidija Šuprina
Urednik: Dominik Kenđel
Lektorica: Ana Đorđević

Sveučilište u Zagrebu
Sveučilišni računski centar
Josipa Marohnića 5, 10000 Zagreb
edu@srce.hr

ISBN 978-953-382-005-7 (meki uvez)
ISBN 978-953-382-006-4 (PDF)

Verzija priručnika D351-20221121

Sadržaj

1. dan

Trajanje poglavlja:

10 min

Uvod

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

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

Priručnik se sastoji od 14 poglavlja koja se obrađuju u pet dana, četiri sati dnevno, koliko traje tečaj. Na kraju svakog poglavlja nalaze se vježbe koje će polaznici izvoditi 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.

Uz jednostavne vježbe tijekom tečaja bit će izrađeni i praktični primjeri koji će rabiti baze podataka za obradu podataka, jednostavan adresar s mogućnošću slanja poruka elektroničke pošte i baza evidencije tečajeva.

Ovaj tečaj može poslužiti kao nastavak tečajeva „Uvod u HTML“, koji osigurava potrebno poznavanje osnova jezika HTML, i „Uvod u SQL“, koji omogućava predznanje osnova jezika SQL − koji PHP koristi za obradu podataka u relacijskim bazama podataka. 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 izradu vježbi predviđena je mapa **„…\D351\vjezbe“**, u kojoj se nalaze sve potrebne datoteke. Gotova rješenja svih vježbi nalaze se u mapi **„…\D351\rjesenja“**. U mapi **„..\D351\primjeri“** nalaze se primjeri korišteni u poglavljima, pa ih je moguće i isprobati.

U tečaju se u primjerima koristi baza *tecajevi,* a podaci i struktura baze podataka preuzeti su s tečaja „Uvod u SQL“ autora Edina Mujadževića.

1. Uvod u PHP i MySQL

1. dan

Trajanje poglavlja:

30 min

Po završetku ovoga poglavlja moći ćete:

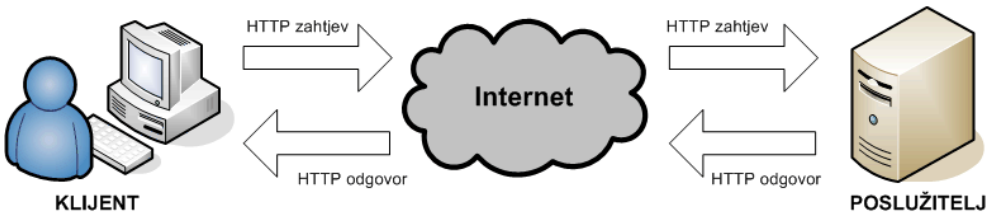
- objasniti što je PHP i MySQL
- definirati klijentsko i poslužiteljsko skriptiranje
- navesti kratak pregled poslužiteljskih tehnologija i baza podataka
- opisati osnovnu sintaksu PHP-a.

PHP (PHP: Hypertext Preprocesor) je skriptni jezik otvorenog kôda namijenjen primarno za razvoj *web*-stranica, ali se može koristiti i za razvoj *desktop* aplikacija. PHP skripte izvršavaju se na poslužitelju, a u kombinaciji s sustavom za upravljanje relacijskim bazama podataka MySQL i *web*-poslužiteljem čini jedno od najpopularnijih rješenja za izradu dinamičkih *web*-stranica i web.

1.1. Klijentsko i poslužiteljsko skriptiranje

Komunikacija na internetu odvija se s dviju strana: klijentske i poslužiteljske (*server*). Uporabom *web*-preglednika klijent šalje zahtjev, npr. adresu neke stranice, a poslužitelj isporučuje odgovor koji sadrži kôd (HTML) tražene stranice (i njemu pridružene datoteke, npr. slike, video itd.), što klijentski preglednik interpretira i prikazuje korisniku kao *web*-stranicu.

Komunikacija između poslužitelja i klijenta odvija se putem HTTP i HTTPS (HTTP secure) protokola, koji se sastoje od HTTP zahtjeva i odgovora. Komunikacija putem interneta dvosmjerna je – i klijent može unutar HTTP zahtjeva poslati neke podatke poslužitelju.



Napomena

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 *web*-stranice

Statička *web*-stranica uvijek će biti ista. Budući da se ne povezuje s bazom podataka i ne ažurira uporabom *web*-aplikacije, sadržaj se prikazuje korisniku točno kako je pohranjen na poslužitelju. Mogućnosti joj se svode na prikazivanje teksta i slika te povezivanje putem linkova s drugim stranicama.

Dinamičke *web*-stranice

Dinamičke web-stranice su *web*-stranice koje se redovito osvježavaju novim podacima, uglavnom povezanim s bazom podataka. Kombiniraju razne skriptne jezike kako bi se na odgovarajući način i u što kraćem vremenu osvježavao i prikazivao novi sadržaj. Omogućuju i personalizirani prikaz određenih komponenata *web*-stranice.

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 šalju se klijentu, a na klijentu se izvršavaju kad budu pozvane (nakon što se stranica učita u preglednik, kad korisnik pritisne tipku i slično). Klijentske skripte nude veliko proširenje mogućnosti *web*-stranica, od čega su možda najkorisniji 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 se skripte, kao i klijentske, nalaze 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 izvršavanja 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 putem *web*-aplikacije.

U tablici se nalazi usporedba skripti koje se izvršavaju na klijentskoj strani i skripti koje se izvršavaju na poslužitelju:

Klijentske skripte	Poslužiteljske skripte
izvršavaju se na klijentu	izvršavaju se na poslužitelju
nekompatibilnost među preglednicima (neki sadržaj 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, pritisak na desnu tipku miša i sl.	aplikacija može reagirati samo na pritisak miša na poveznicu ili na tipku
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

Klijentske i poslužiteljske skripte ne koriste se kao alternativa jedna drugoj, već se programiranjem koriste u kombinaciji.

Osim PHP-a, jedan od najpoznatijih skriptnih jezika je i 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) tehnologija je koja kombinira klijentsko i poslužiteljsko skriptiranje. Na određenu korisnikovu akciju

Napomena

PHP: Hypertext Preprocessor je tzv. rekurzivna skraćenica, kod koje se sama skraćenica pojavljuje u značenju. U svijetu *open source* tehnologija takve su skraćenice česte.

klijentska skripta šalje zahtjev skripti na poslužitelju, koja obavlja obradu podataka iz baze podataka i šalje ih korisnikovu pregledniku bez osvježavanja cijele stranice. Upotrebom tehnologije AJAX *web*-aplikacije postaju brže za uporabu i imaju naprednije korisničko sučelje od aplikacija koje se temelje samo na poslužiteljskim tehnologijama.

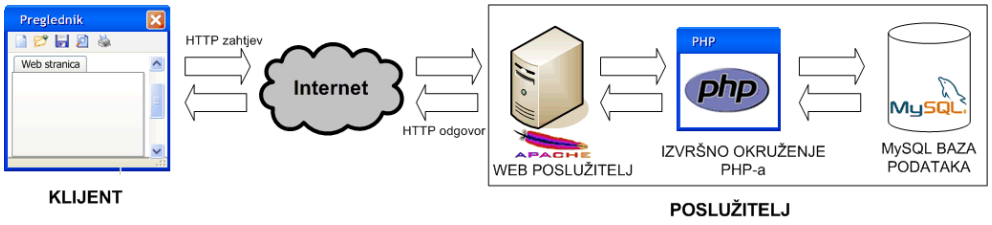
1.2. PHP i ostale poslužiteljske tehnologije

PHP nastao je 1994. kao osobni projekt jednog entuzijasta, a kasnije se u njegov razvoj uključio velik broj programera koji su svi doprinijeli njegovu razvoju. 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. platforme LAMP (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. Ako skripta treba dohvatiti neke podatke iz baze podataka, bazi se postavljaju upiti iz PHP-a. Radi poboljšanja performansi MySQL može biti instaliran na zasebnom poslužitelju. Dobiveni podaci prosljeđuju se PHP-u i PHP ih uklapa unutar HTML kôda. Taj HTML kôd se unutar HTTP odgovora šalje natrag klijentu i podaci se prikazuju u korisnikovu *web*-pregledniku.

Napomena

Posljednja verzija PHP-a je PHP 8.1.5.



Ostale poslužiteljske tehnologije

Konkurent skriptnom jeziku PHP je ASP (Active Server Pages), Microsoftova poslužiteljska tehnologija čije su mogućnosti podjednake. PHP se danas nastavio razvijati i dobivati bolju podršku za objektno orijentirano programiranje. Prednost jezika PHP je i to što se najčešće koristi u kombinaciji s besplatnim platformama na Linuxu, što kod ASP-a nije slučaj jer se koristi u kombinaciji s Microsoftovim platformama.

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 velik broj gotovih kontrola i objekata te činjenica da se njegove datoteke ne moraju prevoditi svaki put kod izvršavanja, već samo prvi put.

Postoji i nekoliko *frameworka* (razvojnih platformi) za PHP (Laravel, CodeIgniter, 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, objekata za rad s bazama podataka i drugih.

Jaka konkurencija je i JSP (JavaServer Pages). Za JSP danas postoji velik broj *frameworka* (Spring, Struts, JavaServer Faces itd.). 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. U posljednje vrijeme vrlo je popularan.

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

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

1.3. MySQL i druge baze podataka

MySQL je najrasprostranjeniji *open source* sustav za upravljanje bazama podataka na *webu*. Njegov razvoj započeo je 1994. godine, a prva verzija objavljena 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 upotrebljava se u zahtjevnim poslovnim okruženjima, pri čemu se prvi značajno koristi i na *webu*.

Napomena

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

Napomena

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.

1.4. Osnove PHP sintakse

PHP naredbe mogu se koristiti u zasebnim PHP skriptama, a 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. Preporučeni način pisanja PHP oznaka je ovaj:

```
<?php

?>
```

Skraćeni način pisanja PHP oznaka izgleda ovako:

```
<?

?>
```

Svaka PHP naredba unutar skripte mora se 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“.

U skripti je moguće uz PHP naredbe pisati i dodatni tekst (komentare) koji se kod pokretanja skripte ne izvodi, a služi za opis ili komentar napisanog kôda. Komentari koji se nalaze u jednoj liniji započinju znakovima ***//*** ili znakom ***#***, a komentari koji se protežu duž jedne ili više linija započinju znakovima ***/****, a završavaju znakovima ****/***.

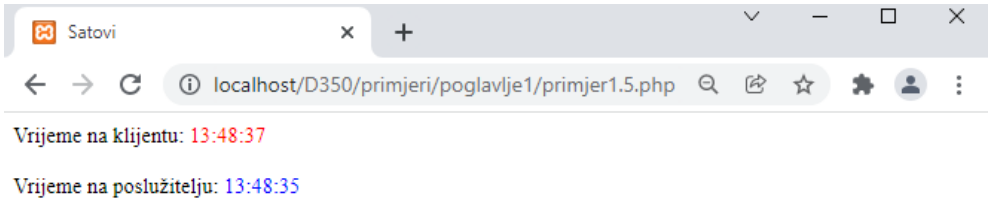
```
<?php
    // tekst koji se ne prikazuje
    echo "Dobar dan";
?>
```

Drugi primjer je:

```
<?php
    /* tekst koji se ne prikazuje */

    echo "Dobar dan";
?>
```

1.5. Razlika između klijentskog i poslužiteljskog skriptiranja

Kao primjer razlike skripte koja se izvršava na poslužitelju i one koja se izvršava na klijentu u pregledniku pokrenut ćemo datoteku *primjer1.5.php*.



Prvi sat prikazuje vrijeme pomoću JavaScripta, koje se stalno ažurira u intervalima od 1 sekunde jer se skripta izvršava na strani klijenta.

Drugi sat prikazuje vrijeme koje ispisuje skripta u PHP-u, koja se izvršava na strani poslužitelja (slanje zahtjeva poslužitelju i primanje odgovora). Da bi se to vrijeme ažuriralo, potrebno je osvježiti stranicu (pritiskom i ili kombinacijom tipki Ctrl + F5)

Vježba 1.1 – Prva PHP skripta

Napomena

Ako ste nakon instalacije XAMPP-a promijenili port iz 80 u neku drugu vrijednost, npr. 8080, upisat ćete adresu „http://localhost:8080/“

1. Pokrenite program Brackets i otvorite datoteku *index.php* iz mape „…\D351\vjezbe\vjezba1.1“.

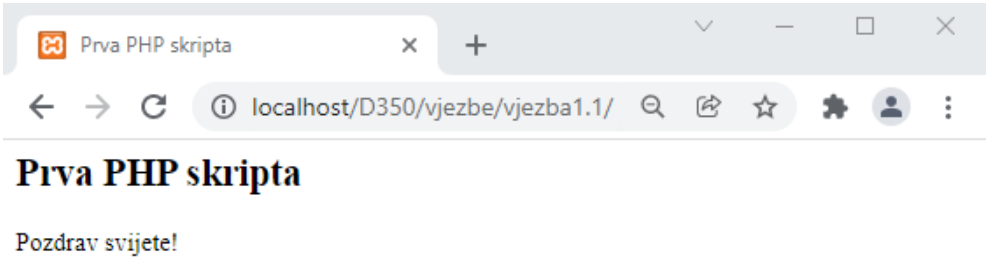
2. Poslije elementa oznake *h2* i prije završne oznake *body* dodajte početnu i završnu oznaku za PHP kôd. Unutar PHP sintakse dodajte naredbu koja će ispisivati tekst „Pozdrav svijete” (masno otisnuti kôd):

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

Spremite promjene u datoteci.

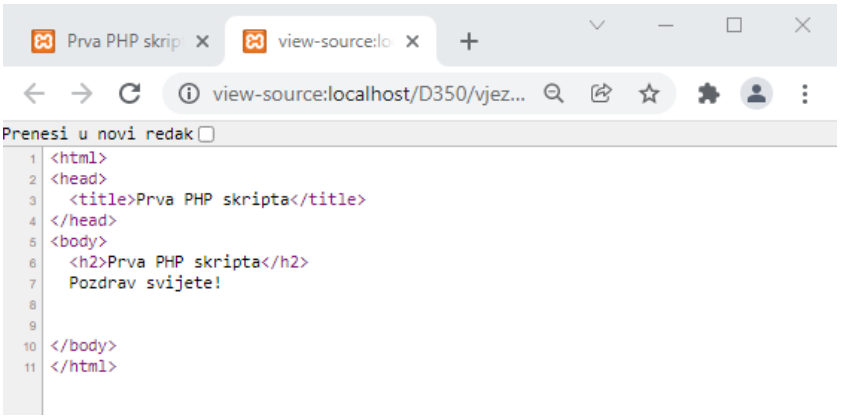
3. Pokrenite servis XAMPP ako nije već pokrenut.

4. U *web*-preglednik upišite adresu:
   http://localhost/D351/vjezbe/vjezba1.1/



5. Uz naslov „Prva PHP skripta“ u pregledniku će se ispisati i tekst „Pozdrav svijete!“.

6. Ako iz izbornika odaberete View→ Source, u novom prozoru prikazat će se kôd stvorene HTML stranice. Možete primijetiti da se unutar HTML kôda ne vide niti PHP oznake niti kôd koji smo postavili unutar njih, već samo tekst „Pozdrav svijete!“. To znači da se u *web*-pregledniku prikazuje gotovi HTML kôd koji se prethodno generira na poslužitelju kad se pozove stranica *index.php*.



2. Obrasci

1. dan

Trajanje poglavlja:
110 min

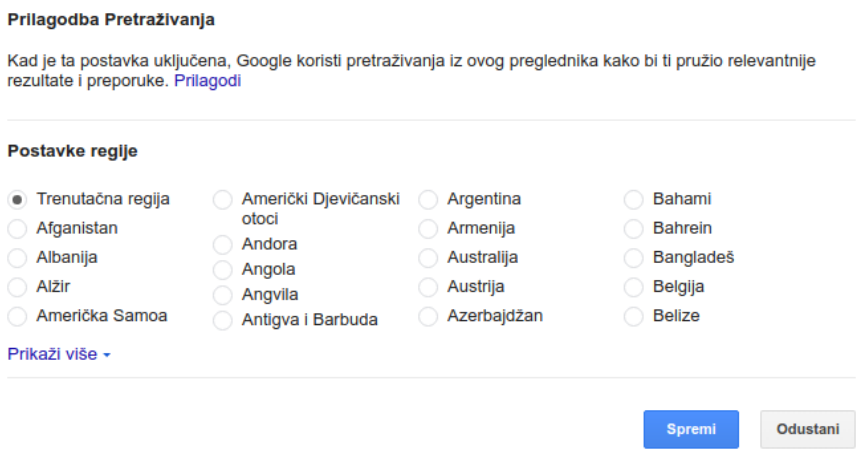
Po završetku ovoga poglavlja moći ćete:

- definirati obrazac i njegove elemente
- izvesti slanje podataka metodama GET i POST
- provesti pristupanje poslanim podacima.

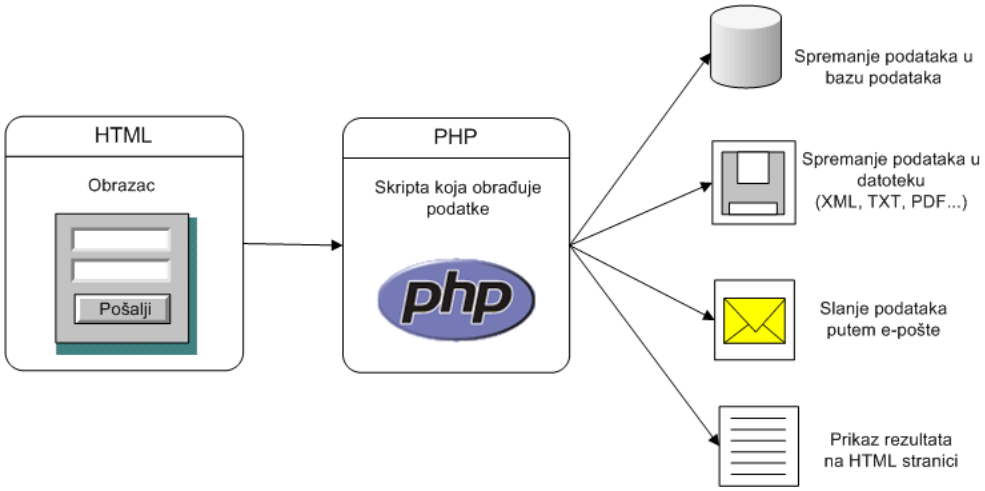
Obrasci služe korisniku za unos, pregled i pohranu podataka u bazu ili datoteku, a mogu se koristiti i samo za prijenos podataka među skriptama.

Obrasci uglavnom sadrže polja za unos teksta, ali mogu sadržavati i interaktivne elemente kao što su padajući izbornik, tipka za odabir i slika.

Slika ispod prikazuje primjer obrasca (podešavanje postavki na tražilici Google).



Kada se otvori stranica koja sadrži obrazac, preglednik prikazuje HTML kôd koji prikaže obrazac. Nakon što korisnik popuni obrazac i pritisne „Pošalji“, podaci se šalju na PHP skriptu koja dalje obrađuje podatke.



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.)

2.1. Elementi obrasca

Napomena

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

Obrazac može sadržavati sljedeće elemente:

- `<input>`
- `<label>`
- `<select>`
- `<textarea>`
- `<button>`
- `<option>`

Oznake elemenata potrebno je postaviti unutar HTML oznake ***form*** koja definira sam obrazac.

Primjer oznake ***form*** mogao bi izgledati ovako:

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

   <!-- ovdje se unose elementi obrasca -->

</form>
```

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 ako se pomoću obrasca šalje datoteka, inače se može izostaviti).

Napomena

Svaki element obrasca mora imati atribut ***name***.

Preporuka je da elementi obrasca sadrže i atribut ***Id*** , preko kojega se kasnije dohvaća vrijednost unesena u obrazac (u PHP skripti koja obrađuje podatke).

Podaci se mogu dohvaćati i preko atributa ***name*** i preko atributa ***class***.

Enctype radi samo uz method="post"

Za unos teksta koristi se element ***<input>***, a kako bi unos teksta unutar elementa bio prikazan, potrebno je dodati postaviti i atribut ***type="text"***.

Polje za unos teksta

Ime osobe:

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

Opcion atributi su:

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

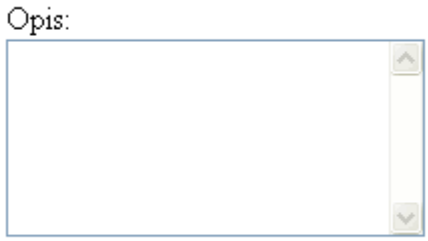
U elementu *<input>* tekst se prikazuje unutar jedne linije.

Polje za unos lozinke


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

Polje za unos lozinke postavlja se elementom ***<input>*** . Ako se želi sakriti prikaz teksta, atribut ***type*** postavlja se na ***password***. Također može imati atribute ***maxlength*** i ***size***.

Višelinijsko polje za unos teksta

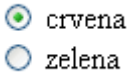



```
Opis:
<textarea name="adresa" cols="40" rows="5">
</textarea>
```

Za unos teksta koji želimo prikazati u više linija koristimo element ***<textarea>***.

Ovom elementu ne može se zadati maksimalna duljina pomoću atributa ***maxlength***. Redovi se određuju pomoću atributa ***rows***, a stupci pomoću atributa ***cols***. Također se može odrediti visina i širina elementa, ali se ona najčešće definira CSS sintaksom.

Tipka za odabir




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

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

Tipka za odabir služi kako bi korisnik mogao odabrati samo jednu od ponuđenih opcija.

Napomena

Atribut „checked“ ne treba imati vrijednost, može samostalno stajati jer je atribut Booleova tipa.

Atribut "selected" ne treba imati vrijednost, može samostalno stajati jer je atribut Booleovog tipa.

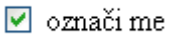
Atribut "multiple" ne treba imati vrijednost, može samostalno stajati jer je atribut Booleovog tipa.

Obično dolazi u grupama, a da bi mogućnosti odabira bile povezane, tj. da se samo jedna tipka može istovremeno pritisnuti, sve tipke moraju imati istu vrijednost atributa ***name***.

Također, da bi skripta koja prima podatke znala koje je tipka pritisnuta, svaka tipka mora imati vlastitu vrijednost atributa ***value***.

Odabiru se može dodati i atribut ***checked***. Ako se atribut ***checked*** postavi na vrijednost ***checked***, tipka će korisniku biti unaprijed odabrana.

Potvrdni okvir




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

Potvrdni okvir također se postavlja uporabom elementa ***<input>***, a koristi se za (isključenje/uključenje) neke mogućnosti ili unos podatka koji potvrđuje je li uvjet istinit ili ne.

Potvrdni okvir može biti unaprijed označen tako da se atribut *checked* postavi na vrijednost *checked*.

Polje za odabir datoteke


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

U obrascu također postoji mogućnost pridruživanja datoteke. To se također postavlja uporabom elementa *<input>* pri čemu se pritiskom na „Choose file.“ (u hrvatskoj inačici preglednika „Odaberi datoteku”) odabere i postavi datoteka koja će se poslati na poslužitelj zajedno s obrascem. Tekst „Choose file…“ ne može se promijeniti jer je definiran jezikom preglednika. Širina elementa može se prilagoditi pomoću atributa *size*.

Padajući izbornik


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

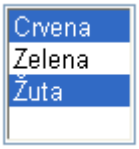
```
          Zelena
     </option>
<option value="0000ff">
          Plava
     </option>

</select>
```

Padajući izbornik postavlja se uporabom elementa ***<select>***.

Vrijednosti odabira postavljaju se unutar oznaka ***option***. Ako želimo da podatak koji se šalje 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 istovremeno




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

Za ovaj izbornik također se koristi element ***<select>*** samo što ovdje korisnik prilikom odabira može odabrati više od jedne ponuđene vrijednosti, uporabom tipke *Ctrl*.

Mogućnost višestrukog odabira dobiva se tako da se oznaci ***select*** postavi atribut ***multiple*** na vrijednost ***multiple***, a pojedine se vrijednosti također upisuju unutar oznaka ***option***.

Tipka


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

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

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

Za obradu podataka u obrascu koristi se tipka za slanje podataka koja se može definirati atributom tipa ***submit***.

Pritiskom na nju upisani podaci šalju se skripti na poslužitelju, koja će ih obraditi. Ako se postavi tipka s atributom tipa ***submit***, postavlja se kao zadana akcija obrascu, što znači da ako pritisnemo **<ENTER>**, učinak će biti isti kao da smo mišem pritisnuli tu tipku. Jedan obrazac (definiran

unutar elementa ***<form>***) ne može imati dvije tipke tipa ***submit***. Ako bi se željelo postaviti više tipki s atributom ***submit***, to se može napraviti postavljanjem više elemenata ***<form>*** unutar jedne PHP skripte.

Tipka s atributom tipa ***reset*** briše sve podatke upisane u obrazac, bez slanja podataka skripti na poslužitelju.

Tipka s atributom tipa ***button*** nema nikakvu predodređenu funkcionalnost, već je njezino ponašanje potrebno posebno programirati (na primjer, povratak na prethodnu stranicu.)

Vrijednost atributa ***value*** označava tekst koji će pisati na tipki.

Slikovna tipka


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

Ako ne želimo koristiti standardnu tipku za obradu podataka, već je želimo zamijeniti npr. slikom, to možemo postići tako što ćemo odabrati element *<input>* i dodati mu atribut *src* s lokacijom slike. Tada će se umjesto tipke prikazati slika čija je lokacija navedena atributu u *src*. Pritiskom na sliku podaci uneseni u obrazac šalju se skripti koja će ih obraditi.

Skriveno polje

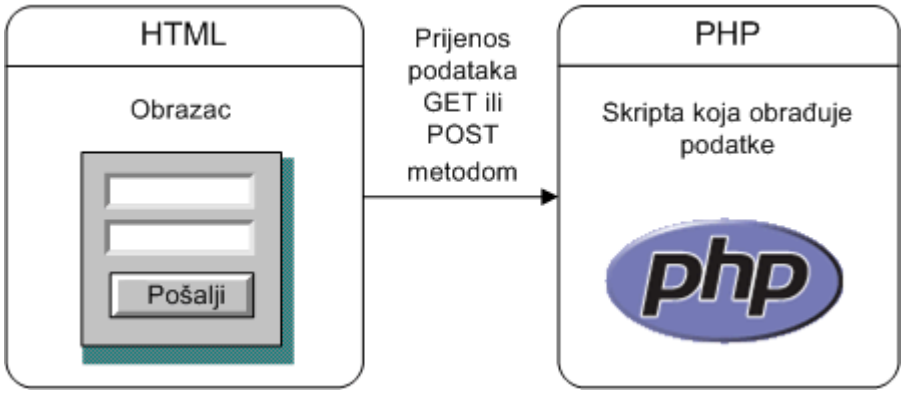
```
<input type="hidden" name="skrivenoPolje" />
```

Ako želimo koristiti mogućnost da se pojedini podaci unose, ali. ne želimo da ih korisnici vide, možemo ih sakriti uporabom elementa *<input>* i atributa *type* kojem ćemo postaviti vrijednost *hidden*. Preglednik ne prikazuje element koji pod atributom *type* sadrži vrijednost *hidden*.

2.2. Metode slanja podataka

Podatke unesene putem obrasca možemo poslati na poslužitelj pritiskom na tipku za slanje podataka (tipka tipa ***submit*** ili ako je postavljena tipka sa slikom). PHP skripta kojoj se podaci šalju navedena je kao vrijednost atributa ***action*** u elemetnu ***<form>***, dok je način prijenosa podataka određen atributom ***method*** u elementu ***<form>***.

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



Metoda GET

Uporabom metode GET podaci koji su upisani u obrazac šalju se 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 metoda GET imat će vrijednost atributa ***method*** postavljenu na ***get***:

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

</form>
```

Prijenos podataka metodom GET koristi se najčešće odvojeno od obrasca kad je potrebno nekoj skripti poslati malu količinu podataka. Većinom se susreće u poveznicama, 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:

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

Uporabom metode POST podaci se šalju unutar tijela HTTP zahtjeva i ne postoji ograničenje na količinu podataka koja se može poslati. Podaci poslani ovom metodom nisu vidljivi u URL adresi skripte.

Napomena

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.

Obrazac kod kojeg se koristi metoda POST imat će vrijednost atributa ***method*** postavljenu na ***post***:

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

</form>
```

2.3. Dohvaćanje podataka unesenih u obrazac

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

Podaci koji se šalju PHP skriptom dohvaćaju se varijablama *$_GET* ili *$_POST*, ovisno o korištenoj metodi prijenosa podataka.

Pretpostavimo da obrazac izgleda kao u ovom primjeru:

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

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

- za metodu GET:

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

- za metodu POST:

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

Osim varijabli *$_GET* i *$_POST*, postoji i varijabla *$_REQUEST* kojom se može pristupiti podacima poslanim i metodom GET i metodom POST. No, ona se rjeđe koristi.

Vježba 2.1 – Obrazac

1. Pokrenite program Brackets i otvorite HTML datoteku pod imenom UnosAdrese.html (u mapi „…\D351\vjezbe\vjezba2.1“).

2. Unutar oznake *body* dodajte oznaku *form.* Kao način prijenosa podataka zadajte metodu POST, a za ime skripte kojoj se podaci šalju navedite „SpremiAdresu.php“.

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

</form>
```

3. Prvi element obrasca bit će ime osobe koja se unosi u adresar. Unutar oznake *form* napišite „Ime:“ i dodajte oznaku *br.*

4. 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/>
```

5. 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 nazivom „adresa“ i nakon nje još dvije oznake *br*.

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

6. 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/>
```

7. Zatim, pomoću tipke 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/>
```

Napomena

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).

8. 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/>
```

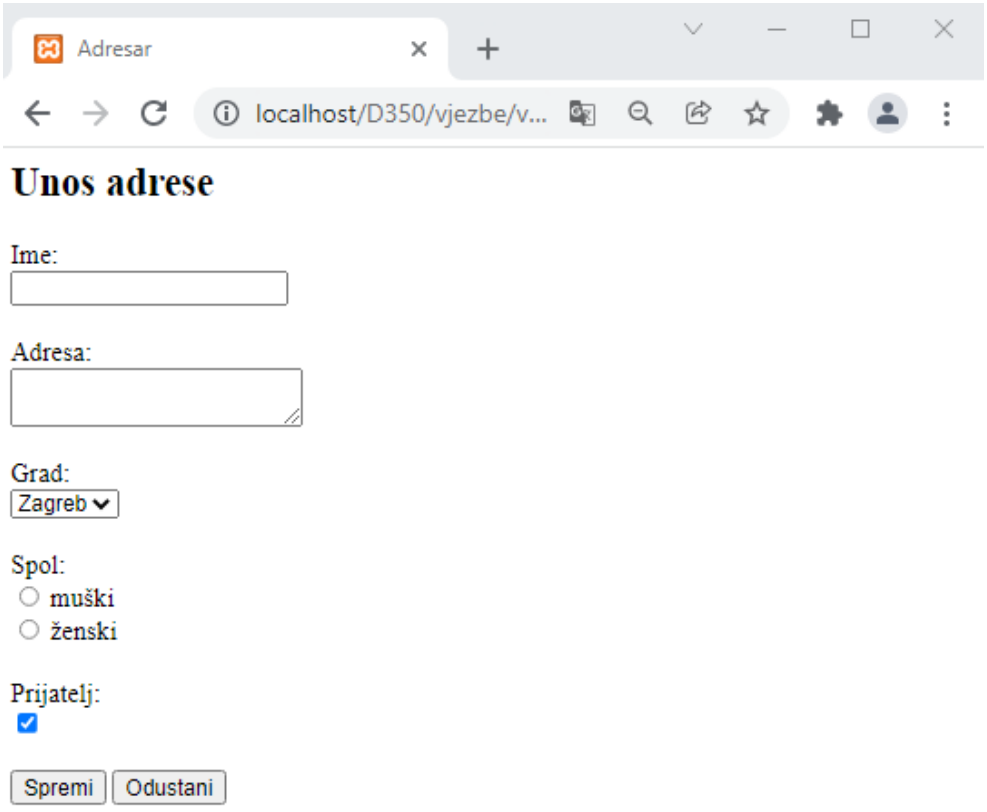
9. Posljednji elementi ovog obrasca su tipka za slanje obrasca i tipka za čišćenje obrasca. Dodajte dvije oznake *input*, tipova *submit* i *reset*. Prvoj tipki kao vrijednost *value* upišite „Spremi“, a drugoj „Odustani“.

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

10. Pokrenite servis XAMPP .

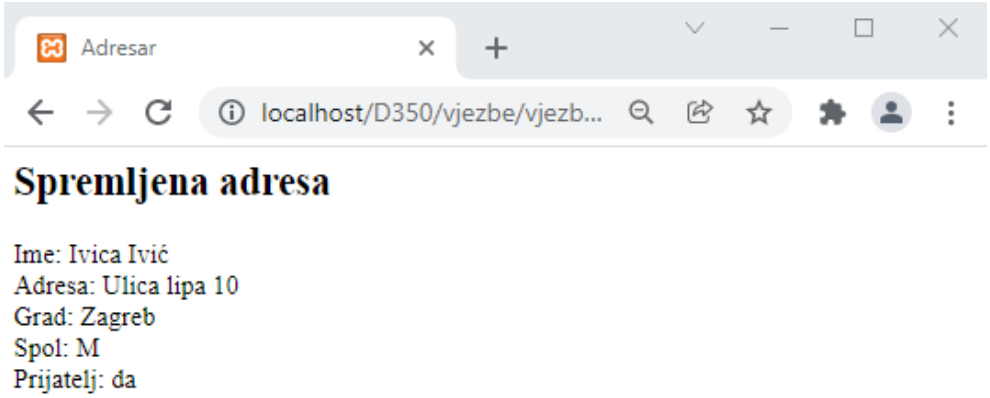
11. Pokrenite svoj *web*-preglednik i upišite adresu http://localhost/D351/vjezbe/vjezba2.1/UnosAdrese.php .

Dobivena stranica trebala bi izgledati kao na slici ispod.



12. Popunite obraza podacima i pritisnite *Spremi*.

13. Tako će se pokrenuti datoteka pod imenom *SpremiAdresu*.php (u mapi „…\D351\vjezbe\vjezba2.1“) koja će na ekranu ispisati podatke što smo ih unijeli u obrazac



Pogledajmo URL adresu u navigacijskoj traci preglednika. Podaci koje smo upisali u obrazac nisu vidljivi kao parametri u URL adresi.

Dodatna vježba – Slanje podataka metodom POST

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

2. Otvorite i datoteku *SpremiAdresu.php* iz prethodne vježbe. Napravite sljedeće izmjene i spremite ih:

- promijenite *$_POST['ime']* u *$_GET['ime']*
- promijenite *$_POST['adresa']* u *$_GET['adresa']*
- promijenite *$_POST['grad']* u *$_GET['grad']*
- promijenite *$_POST['spol']* u *$_GET['spol']*
- promijenite *$_POST['prijatelj']* u *$_GET['prijatelj']*

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

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

3. Rad s MySQL DBMS bazama podataka

1. dan

Trajanje poglavlja:

30 min

Po završetku ovoga poglavlja moći ćete:

- objasniti način rada s aplikacijom phpMyAdmin
- izraditi baze podataka
- izraditi tablice
- prepoznati tipove podataka u sustavu za upravljanje bazama podataka MySQL
- prikazati postavljanje primarnog ključa na tablici
- definirati osnovne upite u jeziku SQL
- spremiti podatake iz baze podataka u datoteku
- izvesti uvoz podataka iz datoteke.

MYSQL je sustav upravljanja relacijskim bazama podataka. Model relacijske baze podataka je model u kojem su podaci spremljeni u više međusobno povezanih tablica (relacija).

Upravljanje podacima iz MySQL sustava obično se izvodi putem naredbene linije unutar samih aplikacija, ali se može upravljati i izravno na samom poslužitelju. Za tu namjenu koristi se aplikacija phpMyAdmin koja obično dolazi uz samu instalaciju MySQL-a, ili se zasebno može skinuti s interneta. Osim aplikacije phpMyAdmin, za rad s MySQL poslužiteljem može se koristiti i aplikacija MySQL Workbench koja se također može skinuti s interneta. Unutar aplikacije MySQL Workbench nalaze se i napredni alati za upravljanje MySQL bazama podataka.

Napomena

Jezik u kojem se prikazuje aplikacija phpMyAdmin postavljen je instalacijom XAMPP aplikacije.

Moguće je promijeniti jezik prikaza, ali treba napomenuti da u pojedinim verzijama nisu podržani pojedini jezici.

Da bi izbjegli različite prikaze na različitim instalacijama, mi ćemo u ovim materijalima koristiti zadani jezik, dakle engleski jezik.

3.1. 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 izvoditi brojne druge akcije potrebne za rad s bazom podataka.

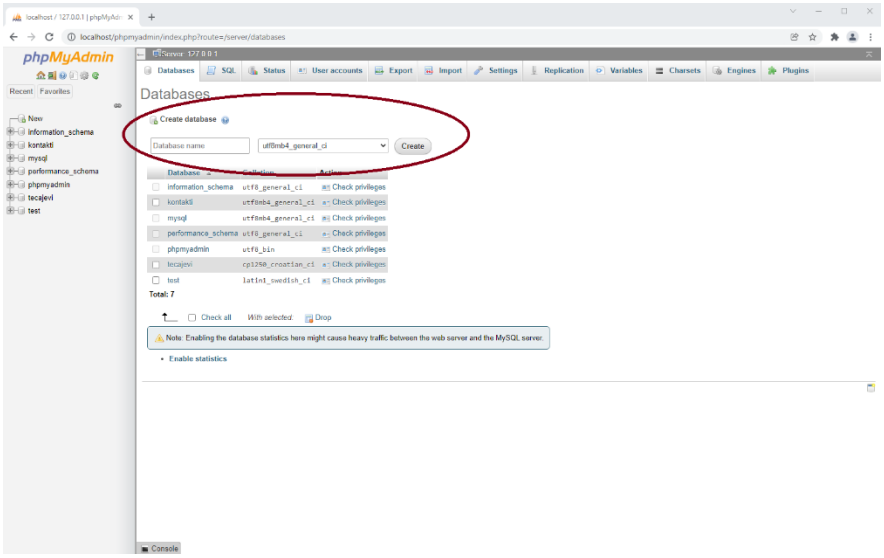
Nakon što se instalira, aplikacija se pokreće u *web*-pregledniku upisivanjem adrese http://localhost/phpMyAdmin/ (ponekad je potrebno dodati na kraju i broj porta, ali to ovisi o samoj instalaciji i podešavanju nakon instalacije).

3.2. Stvaranje baze podataka

Baza podataka može se napraviti pomoću ove SQL naredbe (ako upravljanje MySQL-om izvodi putem naredbenolinijskog alata):

```
CREATE DATABASE imeBaze
```

U aplikaciji phpMyAdmin također se može stvoriti nova baza podataka. Na početnoj stranici aplikacije nalazi se dio s naslovom „phpMyAdmin“ u kojem se, zajedno s poveznicama na baze podataka, nalazi obrazac za stvaranje nove baze podataka, Potrebno je upisati naziv nove baze i pritisnuti na *Stvori* (Create).



Prilikom stvaranja nove baze podataka potrebno je odabrati *collation,* odnosno način abecednog sortiranja znakovnih nizova (koji se razlikuje kod različitih jezika). Odabirom *collationa* u aplikaciji phpMyAdmin odabire se i odgovarajuća kôdna stranica, što određuje na koji će se način 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.

3.3. Stvaranje tablice

Osnova svake relacijske baze podataka su tablice vezane relacijama. U tablicu se upisuju podaci. Entiteti su objekti o kojima prikupljamo podatke. Za svaku vrstu entiteta koji će se pohranjivati u bazu podataka stvara se odgovarajuća tablica koja će sadržavati podatke o entitetu.

Tablice su definirane nazivom, poljima i tipovima podataka što ih sadrže. Svaki zapis koji se sprema u bazu podataka zapisuje se kao redak u odgovarajuću tablicu.

Za jednostavan adresar koji je razrađivan u vježbama uz prethodna poglavlja može se definirati tablica *kontakt*. Kontakti se mogu zapisivati u tablicu prema ovom primjeru:

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 putem više polja, a kombinacija vrijednosti tih polja pritom mora biti jedinstvena za svaki redak.

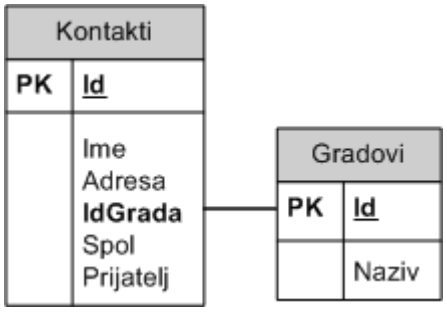
U dizajnu baza podataka teži se tome da se podaci ne ponavljaju na više mjesta. Jedan podatak trebao bi se nalaziti samo na jednom mjestu (radi uštede diskovnog prostora, ali i zbog toga što se u slučaju promjene promjena podatka treba izvršiti samo na jednom mjestu). U gornjem primjeru nazivi gradova bit će isti kod velikog broja kontakata. Stoga bi se grad mogao izdvojiti kao zaseban podatak 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

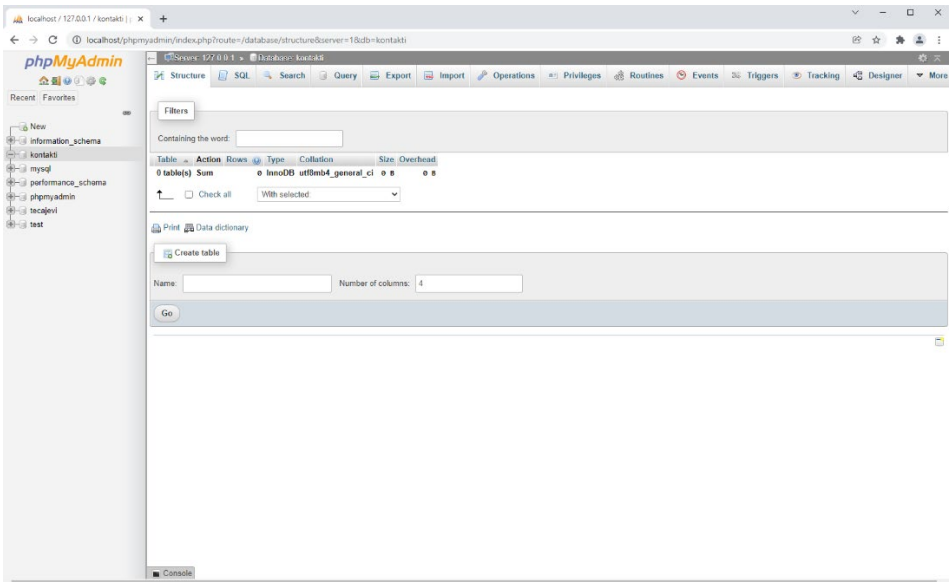
Tablica s kontaktima i tablica s gradovima sad su povezane poljem *IdGrada*, a nazivi gradova nalaze se u zasebnoj tablici. Dijagram ove baze podataka izgledat će ovako:



Nova tablica stvara se pomoću naredbe CREATE TABLE (u naredbenolinijskom alatu), 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 putem 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 te pritisnuti *Go* .



3.4. Tipovi podataka

Sljedeći korak u stvaranju tablice je definiranje tipova podataka koje će pojedina polja koristiti. Pregled tipova podataka koje koristi MySQL dan je u sljedećoj tablici:

Tip podatka	Opis
INT	Cjelobrojni srednje veličine: od -2 147 483 648 do 2 147 483 647.
TINYINT	Vrlo malen cijeli broj: od -128 do 127)
SMALLINT	Malen cijeli broj: od -32 768 do 32 767
MEDIUMINT	Srednje velik cijeli broj: od -8 388 608 do 8 388 607
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): od 01. 01.1000 do 31.12. 9999.
DATETIME	datum i vrijeme (format YYYY-MM-DD HH:MM:SS): od 1000-01. 01. 1000. 00:00:00 do 31. 12. 9999 23:59:59
TIMESTAMP	vremenska oznaka (format YYYY-MM-DD HH:MM:SS) − označava početak Unix epohe (01.01.1970.)
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 0-65535)
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, phpMyAdmin će postaviti predefiniranu 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*.

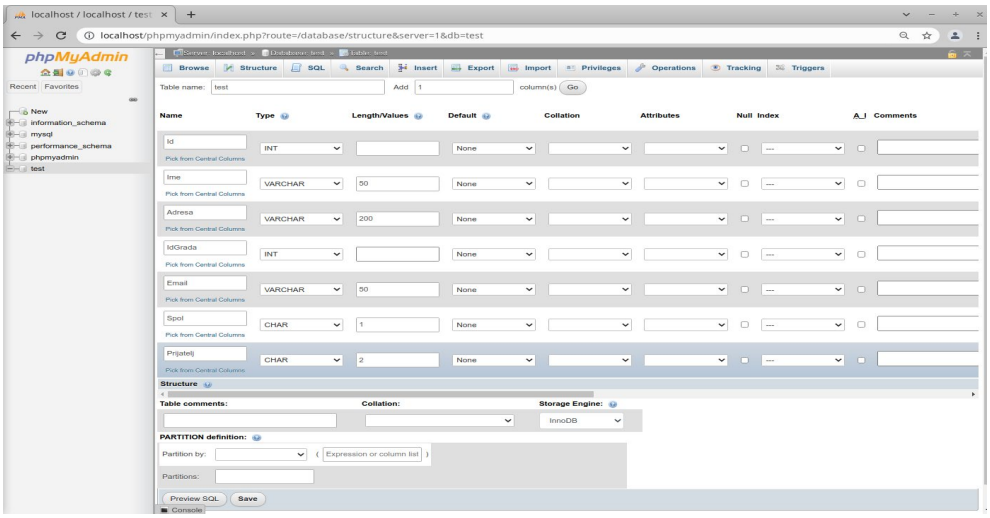
Također je moguće zadati predefiniranu vrijednost polja korištenjem ključne riječi *DEFAULT*.

Primarni ključ postavlja se navođenjem ključnih riječi PRIMARY KEY na kraju definicije polja.

Naredba kojom bi se definirala tablica *kontakti* iz prethodnog primjera izgledala bi ovako (u naredbenolinijskom alatu):

```
  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)
);
```

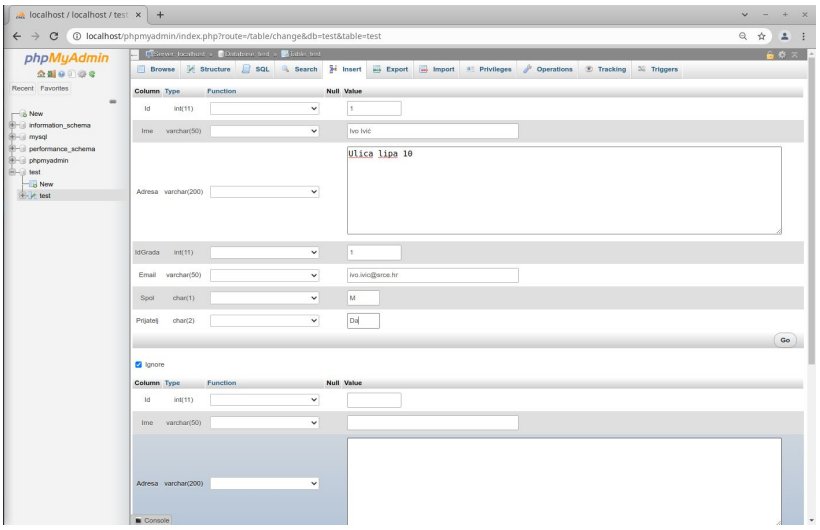
Definiranje polja u aplikaciji phpMyAdmin izgleda ovako (nakon što se upišu naziv tablice i broj polja):



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, a u polju za unos u stupcu *Default* moguće je unijeti predefiniranu vrijednost za polje.

Odabirom indeksa Primary može se postaviti primarni ključ na tablicu, a postavljanjem kvačice na *AI* može se odabrati automatski unos podataka.

Pritiskom na *Save* (Spremi) tablica će biti stvorena i bit će vidljiva u popisu tablica u lijevom okviru aplikacije.



3.5. Dodavanje, izmjena i brisanje redaka

Upisivanje novih redaka u tablicu, ako se ne izvodi putem korisničke aplikacije, može se obaviti putem aplikacije phpMyAdmin. U lijevom okviru aplikacije potrebno je odabrati željenu tablicu i pritisnuti na poveznicu *Insert*.

Potrebno je jednostavno upisati željene vrijednosti za svako polje. Aplikacija nudi mogućnost dodavanja dvaju redaka odjednom (u tom

slučaju polje za označavanje *Ignore* ne treba biti označeno). Za spremanje upisanih podataka pritisnite tipku *Go*.

Za izmjenu ili brisanje retka potrebno je pritisnuti na *Browse* i u željenom retku odabrati ikonu za izmjenu ( ) ili brisanje ( )

Dodavanje, izmjena i brisanje redaka mogući su i pomoću SQL naredbi u aplikaciji ili u naredbenolinijskom alatu, što će biti obrađeno u sljedećem potpoglavlju.

3.6. Izvođenje upita nad bazom podataka

U jeziku SQL možemo koristiti sljedeće naredbe – SELECT, INSERT, UPDATE i DELETE.

SELECT služi za dohvaćanje podataka iz tablice. Izraz koji bi dohvatio sve podatke iz tablice *kontakti* izgleda ovako:

```
SELECT * FROM kontakti
```

Zvjezdica nakon naredbe SELECT označava da se iz tablice *kontakti* čitaju sva polja. Izraz koji dohvaća samo neka polja izgleda ovako:

```
SELECT Ime, Adresa, IdGrada, Email FROM kontakti
```

Svaki izraz 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 izraz dohvatit će sve kontakte koji su iz Zagreba (identifikator grada 1) i vode se kao prijatelji.

Dodavanje novog retka vrši se pomoću naredbe INSERT. Izraz kojim se dodaje novi redak u tablicu *kontakti* izgleda ovako:

```
INSERT INTO kontakti
  (Id, Ime, Adresa, IdGrada, Email, Spol, Prijatelj)
  VALUES ( 2, 'Tom Tomić', 'Aleja jorgovana 5', 2,
  'ttomic@srce.hr', 'M', 'Da' )
```

Izmjena retka vrši se pomoću naredbe UPDATE:

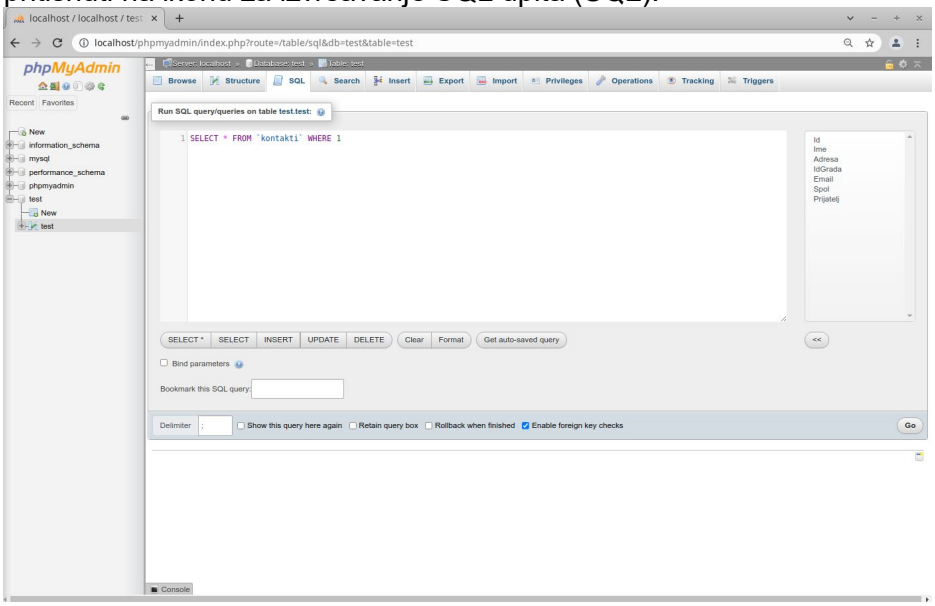
```
UPDATE kontakti SET IdGrada = 1
  WHERE Id = 2
```

Brisanje retka vrši se pomoću naredbe DELETE:

```
DELETE from kontakti WHERE Id = 2
```

Naredbe UPDATE i DELETE 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 izraz putem aplikacije phpMyAdmin, potrebno je najprije odabrati željenu bazu podataka i potom, u lijevom okviru aplikacije, pritisnuti na ikonu za izvršavanje SQL upita (*SQL*).



U novom prozoru koji će se otvoriti potrebno je upisati tekst upita i pritisnuti *Go*.

3.7. 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 naredbu ***Export***. Pod opcijom *Format* moguće je odabrati vrstu datoteke u koju će se podaci spremiti – SQL datoteku (niz naredbi CREATE TABLE i INSERT čijim se izvršavanjem baza podataka može obnoviti), CSV datoteku (vrijednosti odvojene zarezom), XML datoteku ili druge formate.

Pritiskom na *Kreni* izvršit će se izvoz podataka u određenu mapu ili automatski u mapu Preuzimanja (zavisno o postavkama preglednika).

Napomena

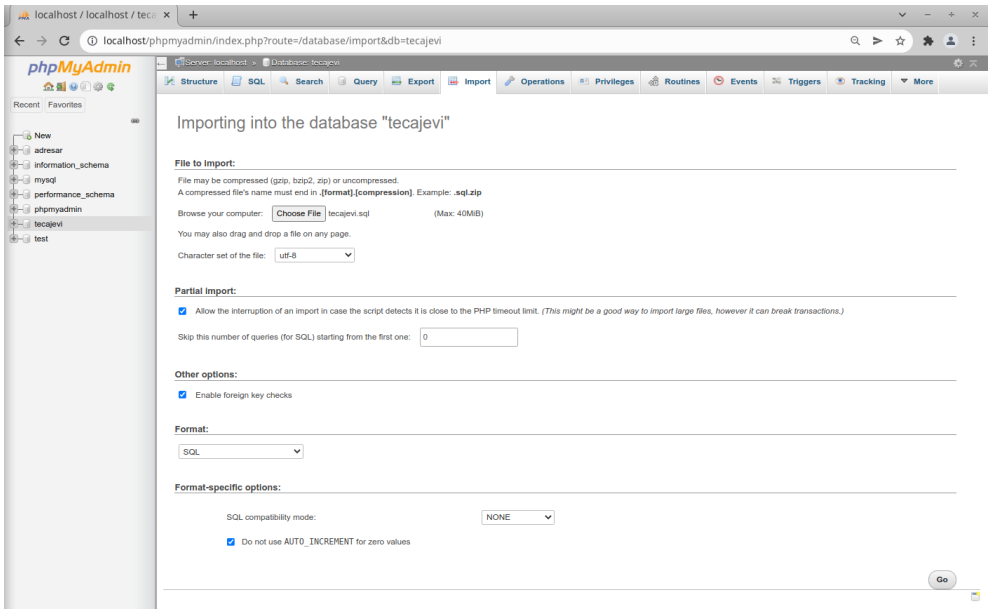
Podaci u bazi podataka *tecajevi* preuzeti su iz baze podataka koja se koristi na tečaju D301 - Uvod u SQL autora Edina Mujadževića

3.8. Uvoz podataka

U MySQL bazu podatke možemo uvesti iz datoteka. Za tu namjenu u aplikaciji phpMyAdmin koristi se naredba ***Import***.

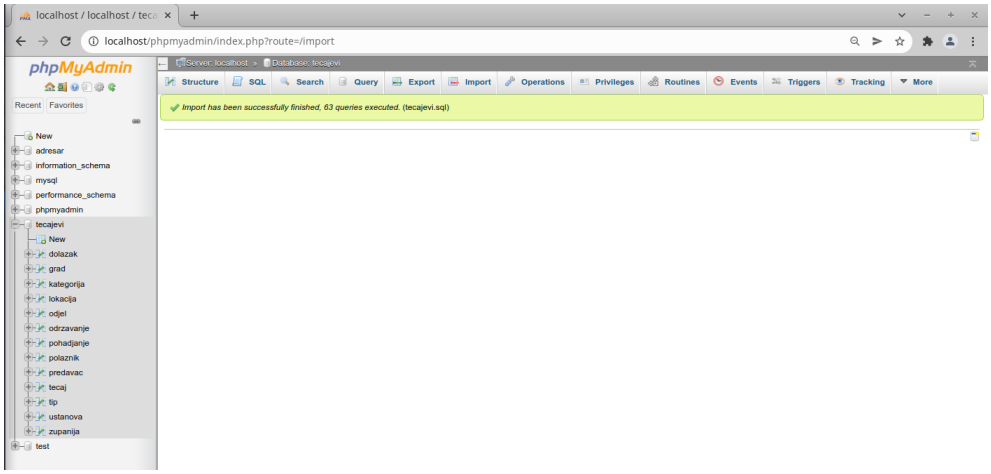
Da bismo uvezli podatke iz datoteke u neku bazu, uvjet je da ta baza već postoji u aplikaciji phpMyAdmin, tj. na MySQL poslužitelju.

Kada odaberemo naredbu ***Import,*** pritiskom na ***Odaberi datoteku (Choose File)*** postavimo datoteku iz koje će se uvesti podaci. Ekstenzija datoteke iz koje se najčešće uvoze podaci je SQL, ali moguće je uvesti podatke i iz **CSV** i **XML** datoteka.



Kada smo uključili u odabir datoteku iz koje želimo uzeti podatke, odaberemo ***Go***.

Nakon toga se automatski obavlja uvoz podataka i, ako nema pogrešaka prilikom uvoza, dobivamo poruku o uspješno uvezenim podacima:



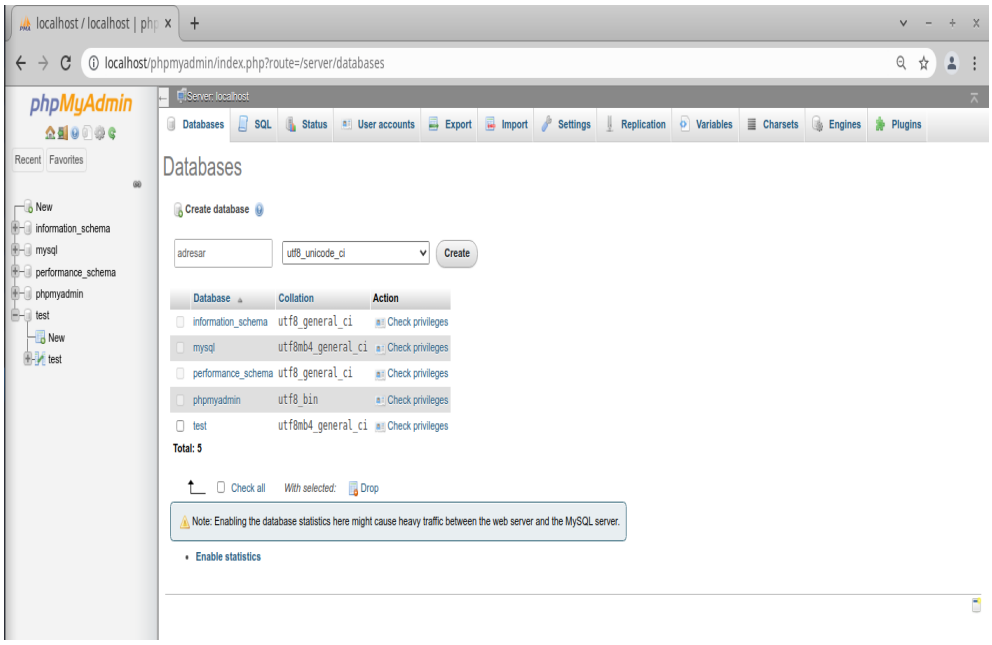
Vježba 3.1 – Stvaranje baze podataka pomoću aplikacije phpMyAdmin

U vježbi se stvara baza podataka ***adresar*** koja će se upotrebljavati za spremanje podataka.

1. Pokrenite servis XAMPP.

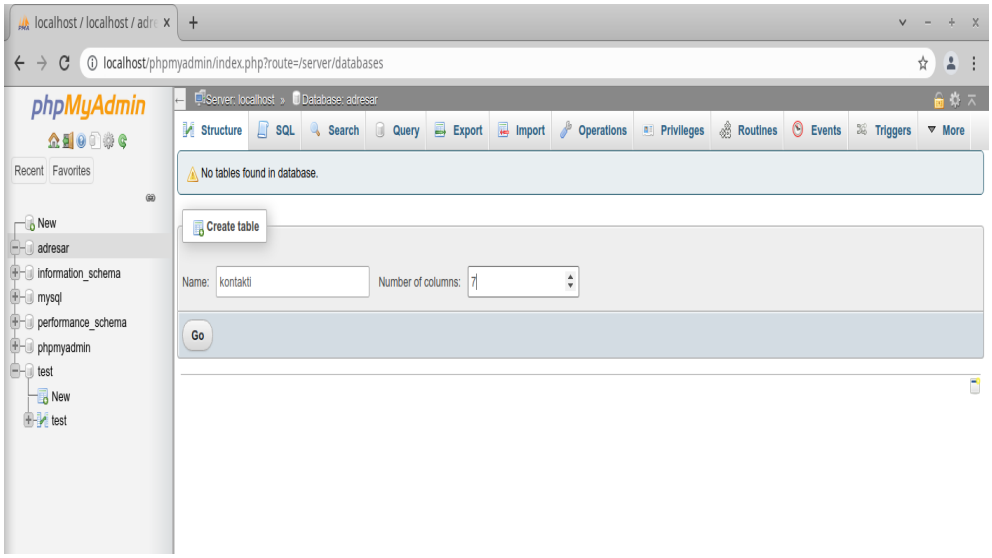
2. Pokrenite aplikaciju phpMyAdmin tako da u *web*-preglednik upišete adresu http://localhost/phpmyadmin/.

3. Stvorite novu bazu podataka. Upišite ime baze u polje za unos teksta (ispod natpisa *Create database*). Baza će se zvati ***adresar***. Za *collation* baze odaberite vrijednost *utf8_general_ci*. Pritisnite tipku *Create*.



Ako je sve u redu, prikazat će se poruka *Database has been created*.

4. Stvorite novu tablicu u bazi podataka. Ispod natpisa *Create table* upišite ime tablice te pritisnite tipku *Go*.



5. 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	1
Prijatelj	CHAR	1

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 tako će 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. Budući da 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 kao maksimalnu duljinu zadati 200 znakova.

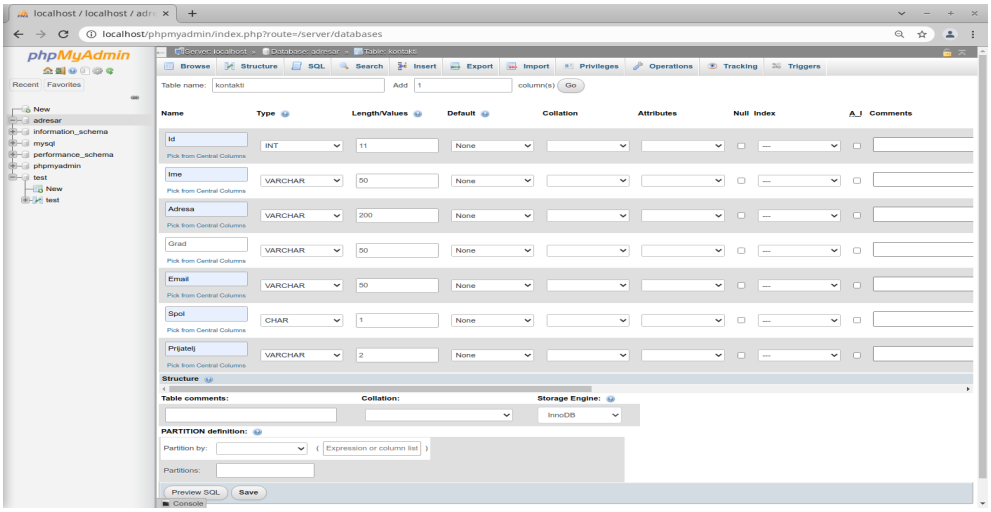
Za polja *Spol* i *Prijatelj* koristit ćemo se tipom podatka CHAR. Podatke za *Spol* zapisivat ćemo kao vrijednosti „M“ za muški i „Ž“ za ženski spol, za što nam trebaju dva znaka (jer je slovo Ž u kôdnoj stranici utf-8 prikazano pomoću dvaju znakova). U polje *Prijatelj* upisat ćemo vrijednosti „Da“ i „Ne“, za što nam također trebaju dva znaka.

Upišite nazive polja iz gornje tablice i odaberite odgovarajući tip podatka u listi za odabir. Za tekstualna polja upišite i odgovarajuću duljinu.

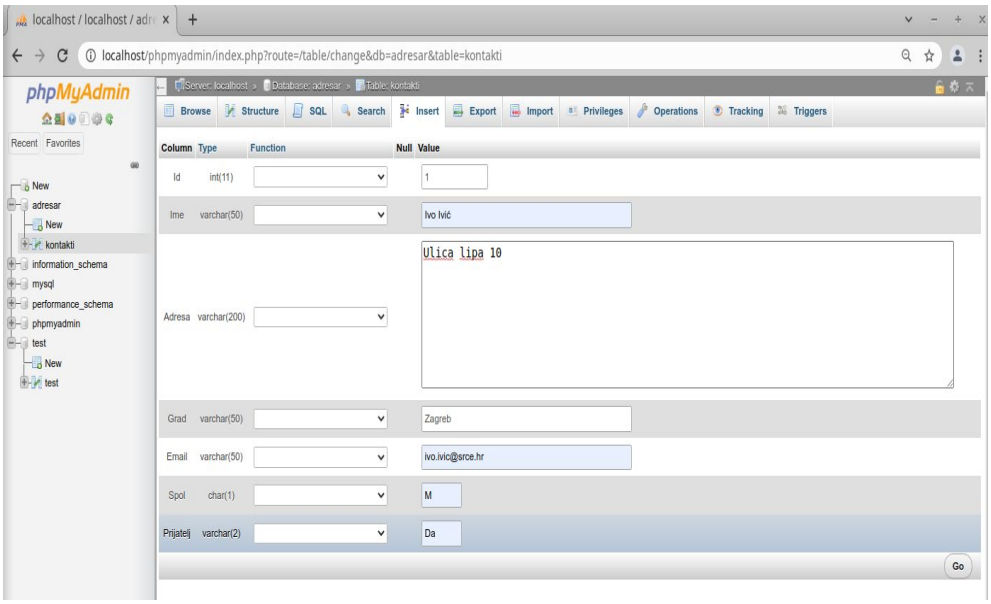
Na polje *Id* postavite primarni ključ odabirom *PRIMARY* pod opcijom *index.*

Također za polje *Id* odaberite *auto_increment* pod opcijom *A_I*. Tako će se vrijednost polja *Id* automatski povećavati za svaki novi redak.

Pritisnite tipku *Save*.

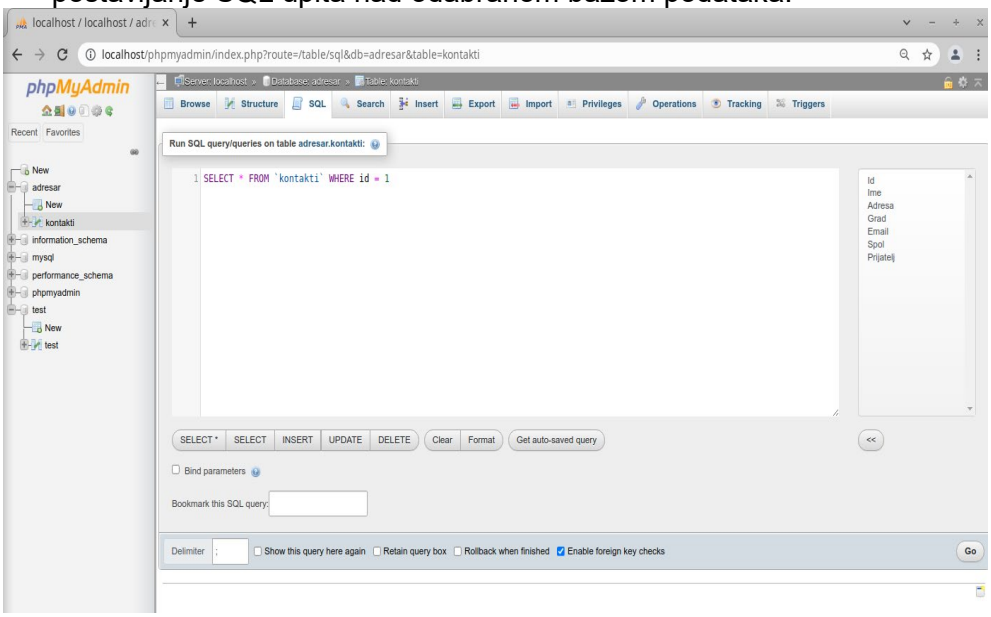


6. Pritisnite naredbu *Insert* (u zaglavlju aplikacije). U stupac *Value* upišite ime, adresu i ostale podatke navedene u slici dolje. Za polje *Id* nemojte ništa upisivati. Pritisnite tipku *Go*.



Primjećujete da je uneseni redak automatski dobio 1 kao vrijednost polja *Id*.

7. Pritisnite na naredbu *SQL* u zaglavlju aplikacije. Otvara se sučelje za postavljanje SQL upita nad odabranom bazom podataka.



Upišite upit „*SELECT * FROM kontakti WHERE Id = 1*“ i pritisnite *Go*.

4. Varijable i operacije nad njima

2. dan

Trajanje poglavlja:

45 min

Po završetku ovoga poglavlja moći ćete:

- definirati varijable i tipove podataka u jeziku PHP
- objasniti pridruživanje vrijednosti varijablama
- opisati primjenu aritmetičkih operatora
- prikazati spajanje znakovnih nizova
- opisati mogućnosti ispisa vrijednosti varijable pomoću naredbe echo.

Osnovna svrha svakog sustava za upravljanje bazama podataka mogućnost obrade i pohrane podataka.

Metode koje se koriste ovise o samom sustavu za upravljanje i verziji PHP-a, a za spremanje i dohvat podataka iz baze koriste se **varijable.**

Za rad s vrijednostima varijabli koriste se **operatori** koji su zapravo simboli računskih operacija koje se izvode nad varijablama (pridruživanje, aritmetičke operacije, operacije usporedbe i drugo).

4.1. 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:

- niz – podatak koji sadrži više povezanih vrijednosti
- objekt – složeni tip podatka koji predstavlja objekt iz stvarnog svijeta (može sadržavati podatke jednostavnog i složenog 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.

Za razliku od većine drugih programskih jezika, varijablu u PHP-u prije uporabe nije potrebno deklarirati, odnosno nije potrebno definirati tip

varijable, već je dovoljno navesti samo njezino ime i pridružiti joj neku vrijednost. Varijabla će sama poprimiti odgovarajući tip podatka ovisno o pridruženoj vrijednosti.

Tijekom uporabe varijable može se promijeniti njen tip, ali to nije preporučeno jer bi moglo dovesti 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.

4.2. Konstante

Vrijednost spremljena u varijabli može se mijenjati tijekom izvođenja programa. **Konstanta** je identifikator, naziv varijable koja se ne mijenja unutar skripte u kojoj je definirana. Može sadržavati vrijednosti kao i varijabla, 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);
  const PI = 3.14;
?>
```

Naziv konstante obično se piše velikim slovima, iako to nije obavezno, ali ovakav način označavanja omogućava lakše razlikovanje konstanti i varijabli. Također, ispred 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.

4.3. Pridruživanje vrijednosti varijabli

Kada varijabli *$varijabla* želimo pridružiti neku vrijednost, pišemo sljedeći izraz:

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

U ovom primjeru pridruživanja znak = je operator pridruživanja. S lijeve strane operatora nalazi se naziv varijable, a s desne strane može se nalaziti konstantna vrijednost, neki izraz (npr. tekstualni, aritmetički), ili se poziva određena funkcija.

U sljedećem primjeru imamo pridruživanje tekstualnih izraza varijabli:

```
<?php
  $ime = "Ana";
  echo $ime;
?>
```

Kada se varijabli pridružuje tekstualni izraz, uvijek se koriste ili jednostruki ili dvostruki navodnici.

Dodjela vrijednosti za cjelobrojnu vrijednost izgleda ovako:

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

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

```
<?php
  $broj = 12.34;
  echo $broj;
?>
```

Moguće je i korištenje eksponencijalnog zapisa:

```
<?php
  $brojeks = 1.2e23;
  echo $brojeks;
?>
```

Varijabla ima vrijednost 1.2 * 10^{23}. Oznaka za eksponent(*e*) može se pisati i malim i velikim slovom.

Vrijednosti varijabli koje sadrže logički tip podataka također nisu osjetljive 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;
?>
```

4.4. 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
  $a = 2;
  $b = 3;
  $zbroj = $a + $b;
?>
```

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

```
<?php
  $a = 2;
  $b = 3;
  $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.

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

```
<?php
  $a = 2;
  $b = 3;

  $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 može se povećati za 1 na ovaj način:

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

Ili se može zamijeniti 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;
```

Napomena

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;
```

```
?>
```

a može se napisati i jednostavnije pomoću operatora **+=** :

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

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`

4.5. Spajanje znakovnih nizova

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

```
<?php
  $ime = "Ivan";
  $prezime = "Ivić";
  echo $ime . $prezime;
?>
```

IvanIvić

Rezultat je ispisani tekst „IvanIvić".

Ako se između imena „Ivan“ i prezimena „Ivić“ želi umetnuti razmak, to se može učiniti ovako:

```
<?php
  echo $ime . " " . $prezime;
?>
```

Ivan Ivić

Kada se koristi kombinacija PHP elemenata s drugim *web*-tehnologijama, ponekad može doći do problema s dvostrukim navodnicima, pa je preporučljivo koristiti apostrofe.

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

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

Dobardan

4.6. Pridruživanje i ispis vrijednosti varijabli iz obrasca

Otvorimo obrazac SpremiAdresu.php u programu Brackets (u mapi „…\D351\primjeri\poglavlje4\“)

Unutar PHP oznaka varijablama ćemo pridružiti vrijednosti kojima će dohvatiti podatke iz obrasca UnosAdrese.php i dodati kôd za ispis podataka:

```
<?php
  $ime = $_POST['ime'];
  $adresa =$_POST['adresa'];

  echo $ime;
  echo "<br/>";
  echo $adresa;

?>
```

Ivan Ivić

Ilica 200

Varijablama ćemo prilikom ispisa pridružiti odgovarajući tekst.

```
<?php
  $ime = $_POST['ime'];
  $adresa = $_POST['adresa'];

  echo "Ime: " . $ime;
  echo "<br/>";
  echo "Adresa: " . $adresa;
?>
```

```
Ime: Ivan Ivić
Adresa: Ilica 200
```

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

```
<?php
 $ime = $_POST['ime'];
 $adresa =$_POST['adresa'];

 echo "Osoba $ime stanuje na adresi $adresa";
?>
```

```
Osoba Ivan Ivić stanuje na adresi Ilica 200.
```

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

```
<?php
 echo 'Osoba $ime stanuje na adresi $adresa';
?>
```

```
Osoba $ime stanuje na adresi $adresa.
```

Napomena

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

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

```
<?php
  $ime = $_POST['ime'];
  $adresa =$_POST['adresa'];
  echo "Osoba {$ime} stanuje na adresi {$adresa}.";
?>
```

```
Osoba Ivan Ivić stanuje na adresi Ilica 200.
```

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:

Napomena

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 (
).

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 \

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

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

Dobar dan.
Hvala što ste došli.

Vježba 4.1 – Rad s varijablama

Pokrenite program Brackets i otvorite datoteku *varijable.php* u mapi „…\D351\vjezbe\vjezba4.1“.

1. U datoteku dodajte naredbe kojima ćete varijabli *$b* dodijeliti vrijednost 15 i ispisati tu vrijednost (masno otisnute naredbe):

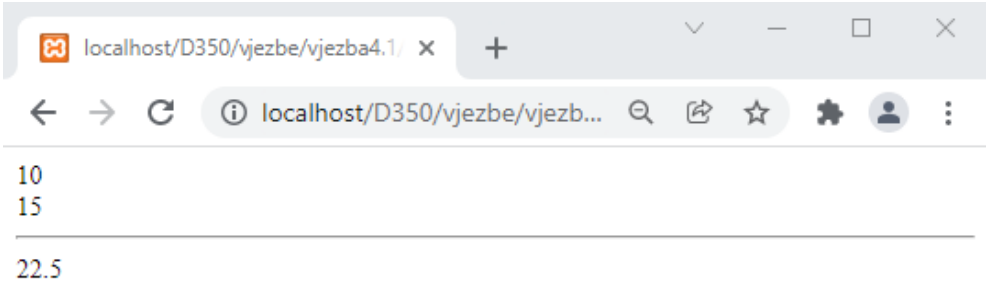
```
<?php

 $a = 10;
 echo $a;
 echo "<br/>";
 $b = 15;
 echo $b;
?>
```

2. Spremite datoteku i otvorite stranicu u *web*-pregledniku (pritiskom na tipku  i kombinacijom tipki Ctrl + F5).

3. Dodajte naredbu koja računa vrijednost varijable $c tako da se vrijednost varijable $c računa kao rezultat izraza 3a – b / 2.

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

4. Spremite datoteku i osvježite stranicu u *web*-pregledniku.

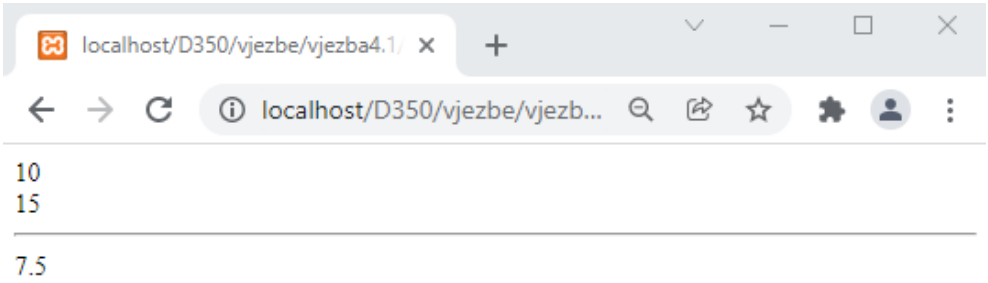


Dobiveni rezultat je 22.5

5. 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;
  ?>
```

6. Spremite datoteku i osvježite stranicu u *web*-pregledniku.



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

Vježba 4.2 – Ispis podataka iz obrasca

1. Pokrenite program Brackets i otvorite datoteku *SpremiAdresu.php* u mapi „…\D351\vjezbe\vjezba4.2“.

2. U datoteku napišite naredbe kojima ćete varijablama pridružiti vrijednosti imena i adrese iz datoteke *UnosAdrese.htm* i koje će iste vrijednosti ispisati na ekranu:

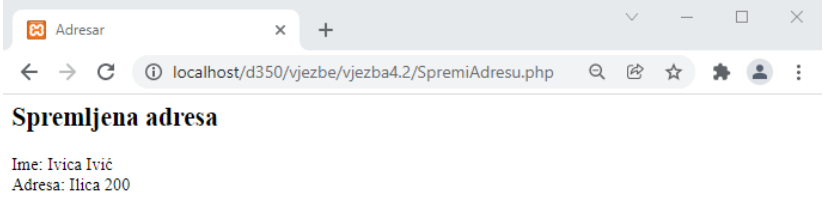
```
<?php

   $ime = $_POST['ime'];
   $adresa = $_POST['adresa'];
   echo "Ime: " . $ime;

   echo "<br/>";
   echo "Adresa: " . $adresa;

?>
```

Izmijenjenu datoteku spremite.

3. Pokrenite servis XAMPP ako nije već pokrenut.

4. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba4.2/UnosAdrese.htm .

5. Unesite proizvoljne vrijednosti u obrazac i pritisnite *Spremi*. Provjerite ispis vrijednosti.



2. dan

Trajanje poglavlja:
45 min

5. Operatori usporedbe, logički operatori i uvjetne strukture

Po završetku ovoga poglavlja moći ćete:

- definirati operatore usporedbe i uspoređivanje varijabli
- razlikovati logičke operatore i logičke izraze
- objasniti jednostavne i složene uvjetne strukture
- opisati ugnježđivanje uvjetnih struktura.

5.1. 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 == $b`	*Vraća TRUE* ako je *$a* jednako *$b*
===	identično	`$a === $b`	*Vraća TRUE* ako je *$a* jednako *$b* i ako su *$a* i *$b* istog tipa podataka
!=	nije jednako	`$a != $b`	*Vraća TRUE* ako *$a* nije jednako *$b*
<>		`$a <> $b`	
!==	nije identično	`$a !== $b`	*Vraća TRUE* ako *$a* nije jednako *$b* ili ako nisu istog tipa podataka
<	manje od	`$a < $b`	*Vraća TRUE* ako je *$a* strogo manje od *$b*
>	više od	`$a > $b`	*Vraća TRUE* ako je *$a* strogo više od *$b*
<=	manje ili jednako	`$a <= $b`	*Vraća TRUE* ako je *$a* manje ili jednako *$b*
>=	više ili jednako	`$a >= $b`	*Vraća TRUE* ako je *$a* više ili jednako *$b*

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 dviju brojčanih 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:

Napomena

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, ali dobiveni rezultat neće biti točan.

Napomena

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*) pretvara se u tekstualni (*string*) po gornjem principu.

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

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

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*.

5.2. Logički operatori

Uz aritmetičke operacije i operacije uspoređivanja, u programskim jezicima postoje i logičke operacije. Operandi, kao i rezultati u logičkim operacijama, logičke su 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	`!$a`	*Vraća TRUE* ako je *$a FALSE*
`&&`	logičko i	2	`$a && $b`	*Vraća TRUE* ako su obje vrijednosti *TRUE*
`and`		4	`$a and $b`	
`||`	logičko ili	3	`$a || $b`	*Vraća TRUE* ako je bar jedna od vrijednosti *TRUE*
`or`		6	`$a or $b`	
`xor`	ekskluzivno ili	5	`$a xor $b`	*Vraća TRUE* ako je samo jedna vrijednost *TRUE*

Operatori && i *and* su 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;          // vraća FALSE (ispisuje "")
  echo $a and $b;    // vraća FALSE (ispisuje "")
  echo $a or $b;     // vraća TRUE (ispisuje "1")
  echo $a xor $b;    // vraća 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
  $a = TRUE;
  $b = FALSE;
  $c = TRUE;
  echo $a and ($b or !$c); //vraća FALSE (ispisuje"")
?>
```

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

5.3. Jednostavne uvjetne strukture

U svim programskim jezicima, pa tako i u skriptnom jeziku PHP, postoje uvjetne strukture koje se koriste za ispis podataka zavisno od postavljenog uvjeta. Ovisno o tome je li neki logički uvjet ispunjen (odnosno je li njegova vrijednost *TRUE* ili *FALSE*) neke naredbe će se izvršiti ili neće.

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*.

```
<?php
  $a = 2;
  $b = 2;
  if ($a > $b)
  {
    echo "a je veće od b.";
  }
?>
```

Napomena

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

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

```
<?php
  $a = 2;
  $b = 3;
  if ($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;
}
```

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

```
<?php
  $a = 2;
  $b = 3;
  if ($a > $b)
  {
    echo "a je veće od b.";
  }
  else
  {
    echo "a je manje ili jednako b.";
  }
?>
```

5.4. Složene uvjetne strukture

U složenijim situacijama moguće je ispitati više uvjeta prije nego što se izvrši neka naredba.

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 uvjet nije ispunjen, izvršava se treća naredba.

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

```
<?php
  $a = 2;
  $b = 3;
  if ($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 blokova *elseif*:

```
<?php
  $a = 2;
  $b = 3;
  if ($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 bloka *else*.

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

Napomena

Bitno je ne izostaviti naredbu *break* na kraju bloka *case*. 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*. Ako odgovarajuća vrijednost nije pronađena, izvršavaju se naredbe navedene nakon ključne riječi *default*.

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

```
<?php
  $a = 2;
  switch ($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. U ovom primjeru izvršit će se ista naredba ako je $a jednako 0 i ako je *$a* = 1:

```
<?php
  $a = 2;
  switch ($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.";
  }
?>
```

5.5. 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 ugniježđena unutar druge. Samo kad je ispunjen prvi uvjet (ako je vrijednost *$pada_kisa* jednaka *TRUE*), izvršit će se provjera drugog uvjeta.

Više ugniježđenih 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 5.1 – Uvjetne strukture

1. Pokrenite program Brackets i otvorite datoteku *index.php* iz mape „…\D351\vjezbe\vjezba5.1“.

2. U datoteku upišite sljedeće naredbe:

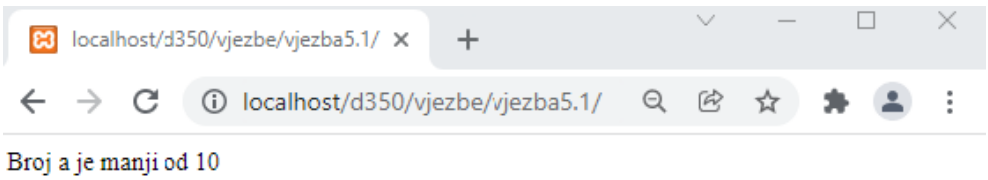
```
<?php
  $a = 1;

  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.

   Spremite izmjene.

3. Pokrenite servis XAMPP ako nije već pokrenut.

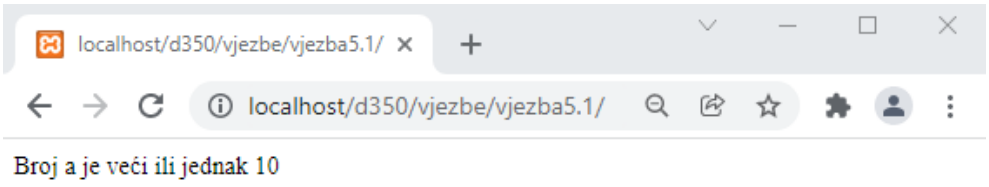
4. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba5.1/ . Očekivano, ispisuje se poruka da je broj a manji od 10.



5. Promijenite *if* strukturu tako da joj dodate i *else* blok koji će ispisivati poruku kad *$a* ne bude manja 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. Osvježite stranicu u *web*-pregledniku. Budući da je vrijednost broja *a* promijenjena, ispisat će se druga poruka:



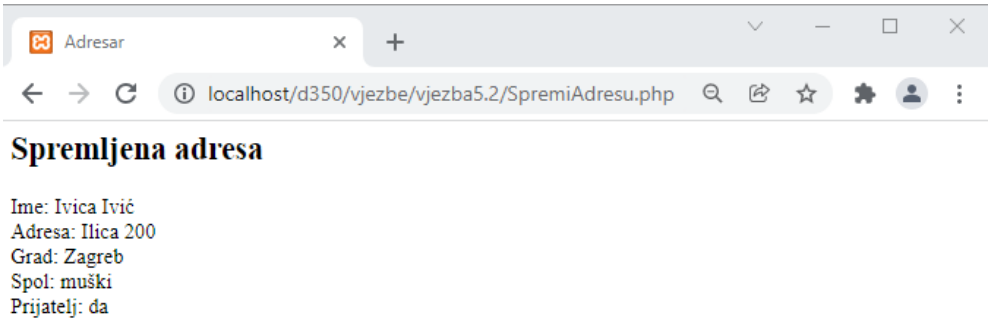
Vježba 5.2 – Uvjetne strukture

1. Pokrenite program Brackets i otvorite datoteku *SpremiAdresu.php* iz mape „…D351\vjezbePHP\vjezbe\vjezba5.2“.

2. Unutar PHP skripte dodajte uvjetnu strukturu koja će temeljem odabira spola u obrascu ispisati tekstualni rezultat:

```
<?php
  echo "Adresa: $adresa <br/>";
      echo "Grad: $grad <br/>";
  if (isset($_POST['spol']))
      {
        if ($_POST['spol'] == "M")
          $spol = "muški";
        else
          $spol = "ženski";
      }
       else
        $spol = "";

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

      echo "Spol: $spol <br/>";
      echo "Prijatelj: $prijatelj <br/>";

?>
```

3. Pokrenite stranicu: http://localhost/D351/Vjezba5.2/UnosAdrese.htm
4. Popunite podatke i pritisnite *Spremi*.
5. Kao rezultat ispisat će vrijednosti iz obrasca:



6. Petlje

2. dan

Trajanje poglavlja:
45 min

Po završetku ovoga poglavlja moći ćete:

- definirati koncept petlje
- objasniti petlju while
- objasniti petlju do...while
- objasniti petlju for
- opisati ugnježđivanje petlji
- prikazati izlazak iz petlje pomoću naredbe break i preskakanje trenutnog kruga petlje pomoću naredbe continue.

Petlje su izrazi kojima se ostvaruje ponavljanje određene naredbe ili bloka naredbi. Dakle, 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.

6.1. 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 navodi se naredba ili naredbe koje će se ponavljati dok god uvjet vrijedi. Kao i kod uvjetnih struktura, moguće je izostaviti vitičaste zagrade ako je riječ samo o jednoj naredbi.

Istinitost uvjeta provjerava se 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 može se 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 je primjeru najprije stvorena varijabla *$i*, kojoj je pridružena početna vrijednost 0. Slijedi petlja čiji je uvjet ponavljanja da je *$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 $i deseti put poveća za 1, njena vrijednost iznosit će 10 i uvjet *$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 ponavljanja uvijek istinit, ova petlja nikada se ne bi prestala izvršavati.

6.2. Petlja *do...while*

Ovaj tip petlje sličan je petlji *while*. Naredbe u tijelu petlje izvršavat će se ako je ispunjen uvjet ponavljanja, s razlikom što 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 nakon 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 svejedno će se izvršiti jedanput:

```
<?php
  $i = 0;
  do
  {
    echo $i . " ";
```

```
    $i++;
  } while($i > 10);
?>
```

```
0
```

6.3. 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 **;** .

- *pocetnilzraz* je početna vrijednost petlje
- *uvjet* je uvjet ponavljanja i provjerava se prije svakog kruga petlje
- *ponavljajuciIzraz* izvršava se nakon svakog kruga petlje.

Svaki od tih triju izraza može se izostaviti. Ako se izostavi *uvjet*, petlja će se izvršavati zauvijek. Izrazi *pocetnilzraz* 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 kao brojač koristi se varijabla (u gornjem primjeru *$i*). U početnom izrazu postavlja se početna vrijednost brojača (0), a u uvjetu konačna vrijednost brojača (10). U ponavljajućem izrazu obavlja se 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.

6.4. Ugnježđivanje petlji

Kao i uvjetne strukture, petlje se mogu ugnježđivati jedna u drugu. Unutrašnja petlja može se promatrati kao zaseban blok kôda. Sljedeći primjer pokazuje da 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 odvojenu točkom.

6.5 Prijevremeni izlazak iz petlje

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

U ovom primjeru petlja bi trebala ispisati brojeve 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 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 tako će biti preskočen ispis broja 6, ali će se svi brojevi nakon njega ispisati.

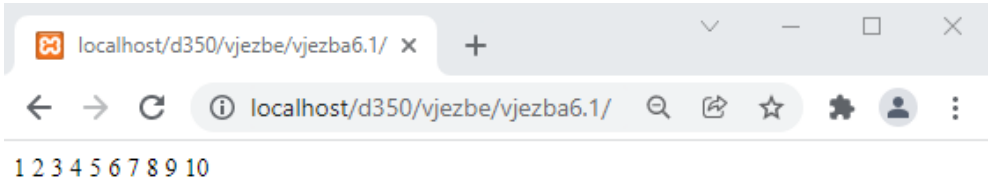
Vježba 6.1 – Korištenje petlje *for*

1. Pokrenite program Brackets i pokrenite stvaranje PHP datoteke.
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 „…\D351\vjezbe\vjezba6.1“ pod nazivom *indeks.php*.

3. Pokrenite servis XAMPP ako već nije pokrenut.
4. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba6.1/



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 . " ";
    }
  }
?>
```

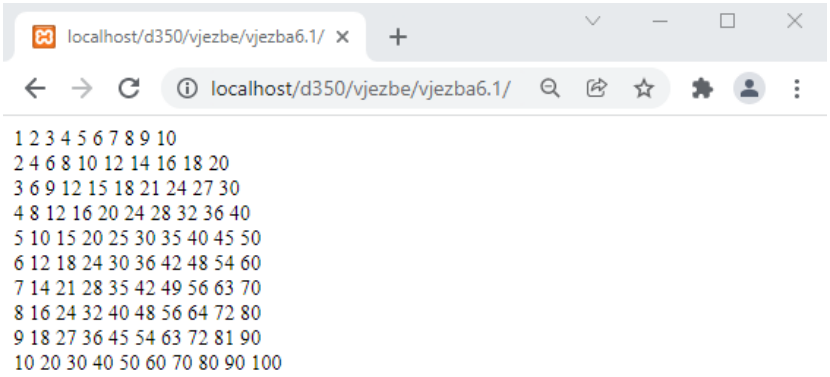
6. U tijelo vanjske petlje dodajte naredbu *echo* koja će ispisati HTML oznaku za novi red:

```
<?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.



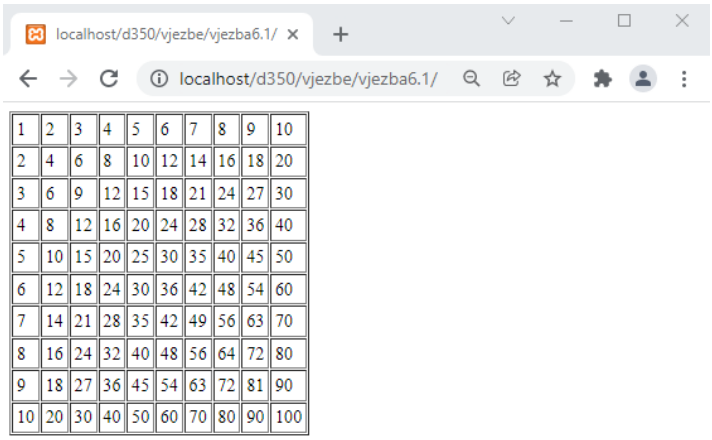
Dobiveni rezultat zapravo je tablica množenja za brojeve od 1 do 10.

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

7. PHP i MySQL

3. dan

Trajanje poglavlja:

90 min

Po završetku ovoga poglavlja moći ćete:

- prikazati povezivanje s bazom bodataka MySQL – funkcija **mysqli_connect**
- objasniti zatvaranje veze – funkcija **mysqli_close**
- opisati izvršavanje SQL upita − funkcija **mysqli_query**
- prikazati dohvaćanje rezultata SELECT upita – funkcija **mysqli_fetch_array**.

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 upit SELECT)
- zatvaranje veze.

7.1. Otvaranje i zatvaranje veze s bazom podataka

Veza ili konekcija je termin koji se koristi u radu s bazama podataka, a označava korisnikov proces kojim se spaja na bazu podataka. Sustav za upravljanje bazama podataka MySQL postavlja predefinirani broj od 151 istovremene konekcije, što znači da korisnik koji se spoji 152. mora čekati da se zatvori neka od prethodno otvorenih veza, odnosno dobiva poruku o preopterećenju. Taj broj može se podesiti unutar konfiguracije MySQL-a.

Preporučuje se da se konekcija zatvori odmah po izvršenju obrade podataka nad MySQL bazom, ali se može prekinuti po izvršenju pojedine skripte.

PHP može koristiti tri načina povezivanja s bazom podataka:

- PDO
- MySQLi (objektno orijentirani)
- MySQLi (proceduralni).

Prihvatljivi su i PDO i MySQLi, samo je razlika što PDO može raditi s 12 različitih sustava baza podataka, dok MySQLi radi samo s MySQL bazama podataka.

PDO primjer komunikacije s bazom:

Napomena

U prijašnjim PHP verzijama koristila se i funkcija ***mysql***, no ona je u novim verzijama zamijenjena sa ***mysqli***.

Napomena

Uporaba "prazne" lozinke nikako nije preporučljiva u stvarnim primjenama jer se podatke uvijek želi zaštititi od neovlaštenog pristupa.

```
<?php
  $servername="poslužitelj";
  $username="korisnik";
  $password="Lozinka";
  $database="Baza";

  try {
  $conn = new PDO("mysql:host=$servername;dbname=$database",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

MySQLi (objektno orijentirani) primjer komunikacije s bazom:

```
<?php
  $servername="poslužitelj";
  $username="korisnik";
  $password="Lozinka";
  $database="Baza";

  // Create connection
$conn = new mysqli($servername, $username, $password,
$database);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

MySQLi (proceduralni) primjer komunikacije s bazom:

```
<?php
  $servername="poslužitelj";
  $username="korisnik";
  $password="Lozinka";
  $database="Baza";

  $veza = mysqli_connect($servername,$username,
  $password,$database);

  if (mysqli_connect_errno())
     {
       echo "Pogreška kod spajanja na poslužitelj: " .
       mysqli_connect_error();
       //exit();
     }
       else
     {
       echo "Spojeni ste na poslužitelj!";
```

```
    }
  mysqli_close($veza);
4?>
```

Za potrebe ovog tečaja mi ćemo koristiti MySQLi proceduralni način povezivanja s bazom.

Za stvaranje veze koristi se funkcija ***mysqli_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 *mysqli_connect* pozivat ćemo ovako:

```
<?php
  $veza = mysqli_connect("localhost", "root", "","test" );

?>
```

Localhost je podrazumijevani naziv lokalnog računala, a korisničko ime „root“ i prazna lozinka preddefinirani su podaci za pristup bazi podataka. Naziv baze podataka na koji se korisnik spaja definiran je varijablom *$database*.

Funkcija vraća identifikator veze koju je otvorila ili logičku vrijednost *false* u slučaju pogreške.

Zatvaranje veze obavlja funkcija ***mysqli_close*** kojoj kao argument zadajemo identifikator na vezu.

Na kraju, može se primijetiti da sve funkcije imaju prefiks *mysqli*, što znači da pripadaju biblioteci funkcija za komunikaciju s bazom podataka MySQL.

Spomenuto je da funkcija *mysqli_connect* vraća *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.

Za primjer ćemo stvoriti novu PHP datoteku *otvoriVezu.php* u mapi „…\D351\primjeri\poglavlje6“ i u datoteku ćemo unijeti kôd koji provjerava je li spajanje na bazu podataka uspjelo:

```
<?php
  $servername="localhost";
  $username="root";
  $password="";
  $database="tecajevi";

  $veza = mysqli_connect($servername,$username,
  $password,$database);

  if (mysqli_connect_errno())
     {
       echo "Pogreška kod spajanja na poslužitelj: " .
       mysqli_connect_error();
       //exit();
     }
       else
     {
```

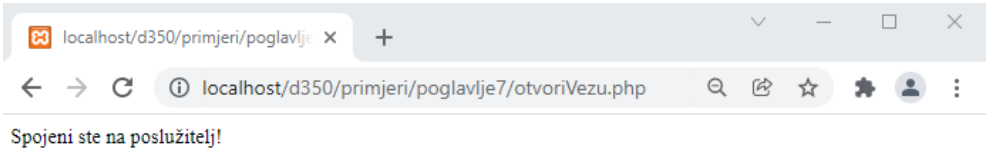
Napomena

Za udaljeno spajanje na poslužitelje obično se koristi IP adresa poslužitelja ili naziv poslužitelja.

Root korisnik rijetko se koristi za pristup MySQL poslužitelju budući da je to glavni administrativni korisnički račun za upravljanje MySQL sustavom, pa samim time predstavlja sigurnosni rizik. Uobičajeno je da se postavi neko drugo korisničko ime i dodijele potrebna prava za pristup.

```
        echo "Spojeni ste na poslužitelj!";
      }
  mysqli_close($veza);
?>
```

U gornjem kôdu otvorit će se veza na MySQL poslužitelj i povezati s bazom podataka. U slučaju da jedan od koraka nije bio uspješan ispisat će se poruka o grešci i prekinut će se izvođenje ostatka kôda pomoću funkcije *exit*.



Vježba 7.1 – Povezivanje na bazu podataka

U ovoj vježbi napravit ćemo povezivanje PHP-a s bazom podataka

1. Pokrenite program Brackets i pokrenite stvaranje PHP datoteke. U datoteku upišite ovaj kôd:

```
<?php

  $servername="localhost";
  $username="root";
  $password="";
  $database="tecajevi";

  $veza = mysqli_connect($servername,$username,
  $password,$database);

  if ($veza)
    {
      echo "Uspješno ste spojeni na poslužitelj!";
    }
    else
    {
      exit("Ne mogu se spojiti na MySql server!");
    }

?>
```

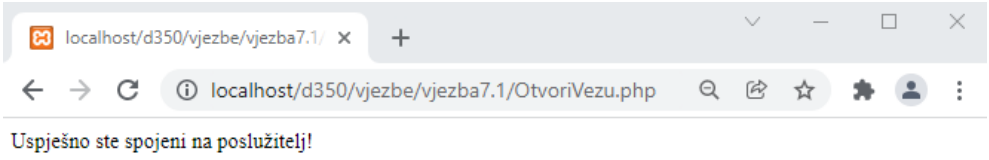
2. Ako jedan od koraka ne uspije, izvršavanje skripte će se prekinuti uz poruku o pogrešci za što se koristi funkcija *exit*. Ako stvaranje veze i odabir baze podataka uspiju, kao rezultat vraća se identifikator veze (varijabla *$veza*).

3. Datoteku spremite u mapu „…\D351\vjezbe\vjezba7.1“ pod nazivom *OtvoriVezu.php*.

4. Pokrenite servis XAMPP ako nije već pokrenut.

5. U pregledniku pokrenite datoteku http://localhost/D351/vjezbe/vjezba7.1.

6. Na ekranu će se prikazati sljedeće:



7.2. Ispis podataka iz baze podataka

Dohvaćanje podataka iz baze podataka ostvaruje se uporabom SQL upita *SELECT*, a koji se poziva funkcijom ***mysqli_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 ***mysqli_fetch_array***, kojoj se kao argument predaje dobiveni identifikator rezultata.

```
<?php
 $rezultat = mysqli_query($veza,"SELECT * FROM polaznik");
 while ($redak = mysqli_fetch_array($rezultat))
 {
    echo $redak['Ime'] . $redak['Adresa'];
    echo $redak['Grad'] . $redak['Email'];
 }
?>
```

Funkcija *mysqli_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 i služi za dohvat podataka za polje *Ime* .

Da bi gornji upit bio uspješno obavljen, potrebno je prije pozivanje funkcije *mysqli_query* obaviti spajanje s bazom podataka, a odmah nakon prestanka čitanja iz baze podataka (nakon izlaska iz petlje) poželjno je zatvoriti vezu s bazom podataka.

Kod upita *SELECT* može se, pomoću funkcije ***mysqli_num_rows***, saznati broj redaka koji zadovoljavaju upit. Ovoj funkciji kao atribut treba predati identifikator rezultata upita.

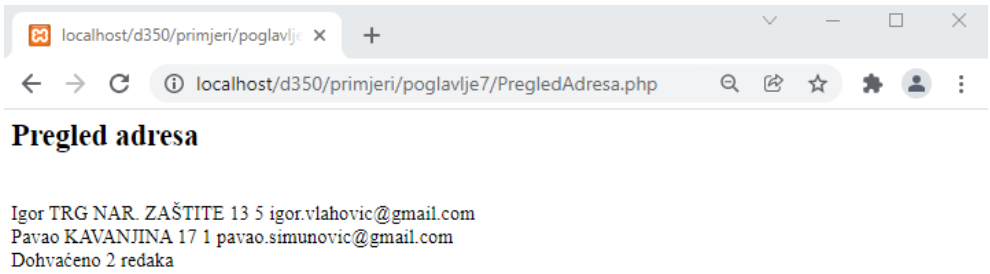
Dobra praksa je osloboditi memoriju zauzetu dohvaćenim podacima (pogotovo ako dohvaćamo veću količinu podataka) čim smo završili s njihovom obradom. Na taj način, slično kao kod zatvaranja veze, oslobađamo resurse poslužitelja za posluživanje drugih korisnika. To se obavlja pozivom funkcije ***mysqli_free_result*** kojoj također predajemo identifikator rezultata.

Za primjer ćemo stvoriti novu PHP datoteku *PregledAdresa.php* u mapi „…\D351\primjeri\poglavlje7“ i u datoteku ćemo unijeti kôd koji će ispisati podatke iz tablice *kontakti*.

Otvaranje veze izvest će se uključivanjem datoteke *otvoriVezu.php*.

Kako bi se ispisali rezultati iz tablice *kontakti*, potrebno je u datoteci *otvoriVezu.php* na kraju isključiti izvršenje funkcije *mysqli_close* jer će se u protivnom zatvoriti veza prije nego što se napravi dohvat podataka iz tablice.

```
<?php
 include ("otvoriVezu.php");
 echo "<br/>";
 $rezultat = mysqli_query($veza,"SELECT * FROM polaznik
order by Id limit 2");
 while ($redak = mysqli_fetch_array($rezultat))
 {
    echo $redak['Ime'] . ' '.$redak['Adresa'].' ';
    echo $redak['GradId'] . ' '.$redak['Email'];
    echo "<br/>";
 }
 echo "Dohvaćeno ".mysqli_num_rows($rezultat)." redaka";
  mysqli_free_result($rezultat);
?>
```



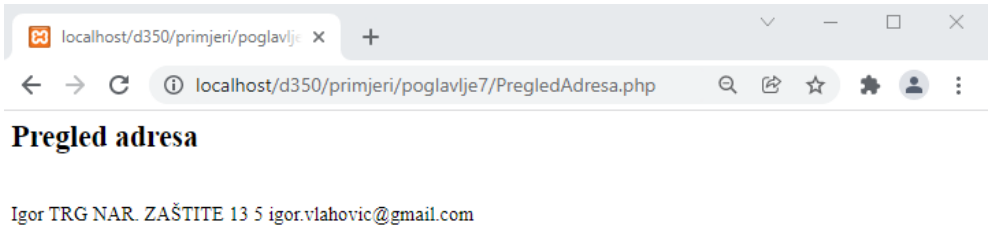
7.3. Ispis podatka koji zadovoljava uvjet

Za ispis podatka koji zadovoljava neki uvjet u SQL upitu dodaje se klauzula *WHERE*.

Dohvat podataka možemo ostvariti uporabom funkcije ***mysqli_fetch_array*** kao u ovom primjeru:

```
<?php
  include ("otvoriVezu.php");
  $rezultat = mysqli_query($veza,
    "SELECT * FROM polaznik WHERE Id = 1");
  if ($redak = mysqli_fetch_array($rezultat))
  {
    echo $redak['Ime'] . ' '.$redak['Adresa'].' ';
    echo $redak['GradId'] . ' '.$redak['Email'];
  }
  mysqli_free_result($rezultat);
?>
```

Rezultat upita bit će jedan redak iz tablice prema uvjetu postavljenom na polje *Id = 1* upisanom u klauzuli *where*.



I u ovom slučaju korisno je osloboditi memoriju poslužitelja pomoću funkcije *mysqli_free_result*.

7.4. Upis podatka u bazu podataka

Upisivanje podatka u bazu podataka vrši se pomoću naredbe INSERT, što se također izvršava pomoću funkcije *mysqli_query*.

Skripta koja će ubaciti novi redak u tablicu *polaznik* izgledat će ovako:

```
<?php

  $sql = "INSERT INTO polaznik (Ime, Adresa, GradId,
        Email, Spol) VALUES ( 'Tom Tomić',
       'Aleja jorgovana 5', 'Split', 'ttomic@srce.hr', 'M')";

   mysqli_query($veza,$sql);
?>
```

Za provjeru uspješnosti upita može se dodati provjera vrijednosti koju vraća funkcija *mysqli_query*.

Za primjer upisa stvorit ćemo novu datoteku *DodajAdrese.php* u mapi „…\D351\primjeri\poglavlje7“, a kôd za unos podatak izgledat će ovako:

```
<?php
  include ("otvoriVezu.php");
  $sql = "INSERT INTO polaznik (Ime, Adresa, GradId,
        Email, Spol) VALUES ( 'Tom Tomić',
```

```
        'Aleja jorgovana 5', 'Split', 'ttomic@srce.hr', 'M')";
  if (mysqli_query($veza,$sql))
  {
    echo "Redak je dodan.";
  }
  else
  {
    echo "Upit nije uspješno obavljen!";
  }
?>
```

7.5. Promjena podatka u bazi podataka

Promjena podataka u bazi podataka izvodi se naredbom UPDATE, a za izvršavanje također se koristi funkcija *mysqli_query*.

Primjer izmjene određenog retka u bazi izgleda ovako:

```
<?php

 mysqli_query($veza,"UPDATE polaznik SET Spol = 'M' WHERE Id
= 2");

?>
```

Napomena

Kod skripti koje rade promjene ili brisanje podataka preporučuje se prethodno dobro provjeriti uvjete koji su postavljeni jer se mogu dogoditi neželjene promjene kada se skripta izvrši ako je bio postavljen pogrešan uvjet.

7.6. Brisanje podatka iz baze podataka

Brisanje podataka iz baze izvodi se naredbom DELETE koja se također izvršava pomoću funkcije *mysqli_query*.

Primjer brisanja retka koji zadovoljava određeni uvjet izgledat će ovako:

```
<?php

  mysqli_query($veza,"DELETE from polaznik WHERE Id = 2");

?>
```

7.7. Funkcije *stripSlashes* i *addSlashes*

Znakovi kao što su \, ', " i još neki mogu izazvati grešku u SQL upitu zbog njihova posebnog značenja u jeziku SQL.

Ovi se znakovi mogu naći u upitu ako ih korisnik postavi 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 upisao ih je korisnik ili su pročitani iz URL-a.

Za uklanjanje (točnije, neutralizaciju) ovih znakova koristi se funkcija *addSlashes*, koja ispred svakog takvog znaka dodaje znak \, 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"]);
 mysqli_query($veza,"UPDATE polaznik SET ime = $ime WHERE Id
= 2");
?>
```

Da bi se korisniku podaci prikazali u istom obliku u kojem ih je upisao u bazu, koristi se funkcija *stripSlashes* koja uklanja dodane znakove \.

```
<?php
  $rezultat = mysqli_query($veza,"SELECT ime FROM
polaznik");
  while ($redak = mysqli_fetch_array($rezultat))
  {
     echo "Ime: " . stripSlashes($redak['ime']) . "<br />";
  }
?>
```

Vježba 7.2 – Ispis podataka iz baze

U ovoj vježbi pomoću PHP skripte ispisat ćemo podatke iz baze

1. U programu Brackets iz mape „…/D351/primjeri/poglavlje7“ otvorite datoteku *PregledAdresa.php*. Unutar oznaka za PHP kôd upišite naredbe koje će izvršiti dohvat 5 podataka iz tablice *polaznik*:

```
<?php
  include("otvoriVezu.php");
  $rezultat = mysqli_query($veza,"SELECT * FROM polaznik
limit 5");
```

Prva naredba je naredba *include* kojom se unutar kôda ove skripte uključuje i sadržaj datoteke o*tvoriVezu.php* čime će veza s bazom podataka otvorena.

Druga naredba izvršit će upit koji vraća sve retke iz tablice *polaznik*, a rezultat upita bit će dostupan putem varijable *$rezultat*.

2. Nakon ove naredbe dodajte sljedeći kôd:

```
        while ($redak = mysqli_fetch_array($rezultat))
        {
          $id = $redak['Id'];
          $ime = $redak['Ime'];
          $prezime = $redak['Prezime'];
          $adresa = $redak['Adresa'];
          $gradId = $redak['GradId'];
          $email = $redak['Email'];
          $spol = $redak['Spol'];

          echo "<tr>";
          echo "<td>$ime</td>";
          echo "<td>$prezime</td>";
          echo "<td>$adresa</td>";
          echo "<td>$gradId</td>";
          echo "<td><a href='UnosPoruke.php?email=$email'>
                $email</a></td>";
          echo "<td>$spol</td>";
          echo "</tr>";
        }
```

Pomoću funkcije *mysqli_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 odmah će potom biti korištene za ispis jednog retka unutar HTML tablice pomoću naredbe *echo*.

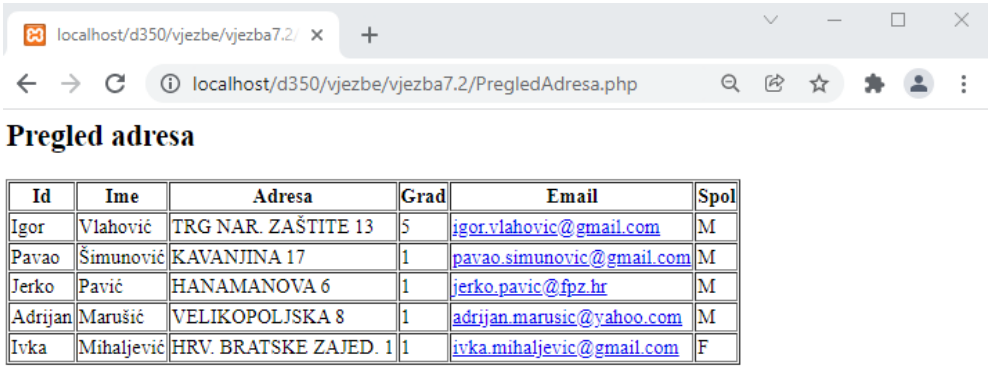
Nakon ove petlje dodajte još i naredbe za oslobođenje rezultata i zatvaranje veze na bazu podataka:

```
        mysqli_free_result($rezultat);
        mysqli_close($veza);
?>
```

3. Spremite datoteku.

4. Pokrenite servis XAMPP ako već nije pokrenut.

5. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba7.2/PregledAdresa.php.



Id	Ime	Adresa	Grad	Email	Spol
Igor	Vlahović	TRG NAR. ZAŠTITE 13	5	igor.vlahovic@gmail.com	M
Pavao	Šimunović	KAVANJINA 17	1	pavao.simunovic@gmail.com	M
Jerko	Pavić	HANAMANOVA 6	1	jerko.pavic@fpz.hr	M
Adrijan	Marušić	VELIKOPOLJSKA 8	1	adrijan.marusic@yahoo.com	M
Ivka	Mihaljević	HRV. BRATSKE ZAJED. 1	1	ivka.mihaljevic@gmail.com	F

Primijetite da je ispisano 5 podataka iz tablice *polaznik* koja se nalazi u bazi *tecajevi*.

Vježba 7.3 – Ispis podatka koji zadovoljavaju uvjet

U ovoj ćemo vježbi u obrascu *PregledAdresa.php* omogućiti da se korisniku, kada pritisne određenu poveznicu u retku u tablici, otvori obrazac *IzmjenaAdrese.php* koji će biti popunjen samo podacima iz tog retka tablice.

1. Otvorite datoteku *PregledAdresa.php* u mapi „…D351/vjezbe/vjezba7.3“ i u njoj napravite sljedeću izmjenu:

```
  echo "<tr>";
  echo "<td>$id</td>";
  echo "<td><a href='IzmjenaAdrese.php?id=$id'>$ime</a></td>";
  echo "<td>$adresa</td>";
  echo "<td>$grad</td>";
```

Ime osobe bit će poveznica koja 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 Brackets otvorite datoteku *IzmjenaAdrese.php*. Ova će datoteka sadržavati obrazac u kojem će se prikazati podaci za odabranog polaznika (čija je vrijednost polja *Id* proslijeđena u URL-u), a ti podaci moći će se i promijeniti.

3. Nakon oznake *h2* ubacite oznake za PHP kôd i sljedeće naredbe unutar njih:

```
<?php
  include("otvoriVezu.php");
    $rezultat = mysqli_query($veza,"SELECT * FROM polaznik
WHERE Id = " . $_GET['id'] );
    while($redak = mysqli_fetch_array($rezultat))
    {
        $id = $redak['Id'];
        $ime = $redak['Ime'];
        $adresa = $redak['Adresa'];
        $grad = $redak['GradId'];
        $email = $redak['Email'];
        $spol = $redak['Spol'];
    }
            mysqli_free_result($rezultat);
            mysqli_close($veza);
?>
```

Najprije se otvara veza na bazu podataka te se zatim izvršava upit *SELECT* koji će vratiti redak čija je 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 prazni rezultat upita i zatvara veza na bazu.

4. U već postojeće elemente za prikaz podataka (*input*) potrebno je u atribut *value* dodati vrijednosti koje su pridružene varijablama, a dohvaćene su iz prethodnog obrasca:

```
<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>
```

5. Na isti način dodat ćemo podatke za listu za odabir grada, samo što će se PHP varijable sada postaviti u atribut <*option*>:

```
 <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>
```

6. Podatak za *email* adresu pridružit ćemo atributu *value*:

```
  <br/><br/>
  E-mail adresa:<br/>
  <input type="text" name="email"
         value="<?php echo $email ?>" >
  <br/><br/>
</form>
```

7. Zatim ćemo tipki za odabir pomoću koje će biti prikazan spol odabranog kontakta pridružiti vrijednosti:

```
  <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>
```

8. Unutar obrasca imamo i skriveno polje putem kojeg će se vrijednost polja *Id* proslijediti skripti *SpremiIzmjenu.php*, koja će spremiti izmijenjene podatke u bazu podataka, pa je i ovdje potrebno dodati vrijednost polja *Id*.

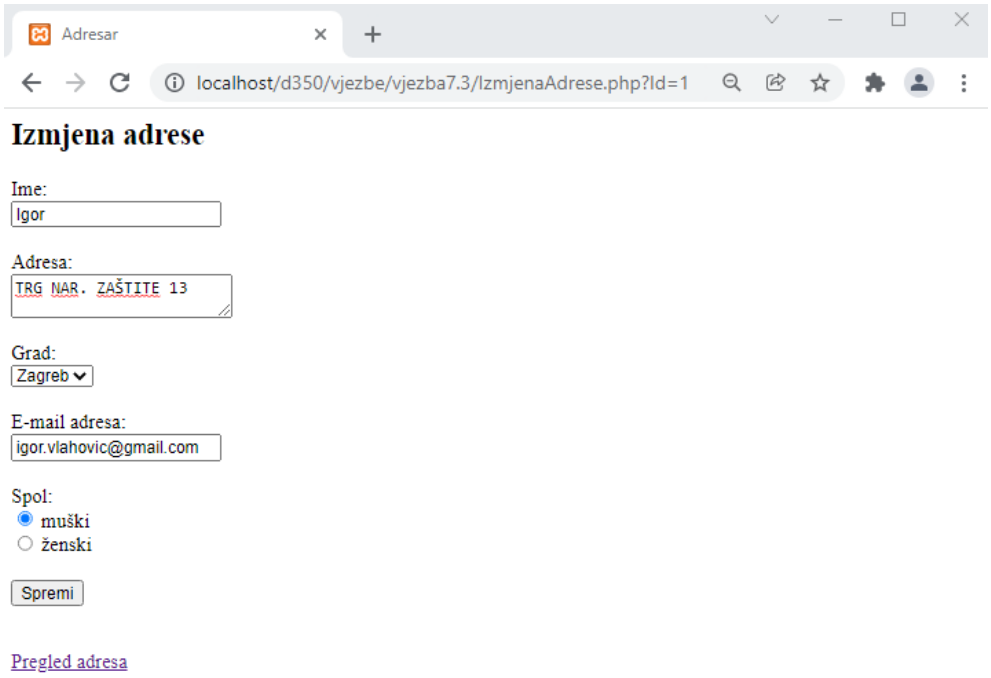
```
  <br/><br/>
  <input type="hidden" name="id" value="<?php echo $id ?>">
</form>
```

Datoteku *IzmjenaAdrese.php* spremite u mapu „…\D351\vjezbe\vjezbe\vjezba7.3“.

6. Pokrenite servis XAMPP ako već nije pokrenut.

7. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba7.3/PregledAdresa.php.

8. Pritisnite na ime prikazanog polaznika.



Poveznica nas je odvela na stranicu *IzmjenaAdrese.php?id=1*. U obrascu su prikazani podaci za redak iz tablice *polaznik* koji ima vrijednost polja *Id* jednaku 1.

Vježba 7.4 – Izmjena podatka u bazi podataka

Napomena

Ova vježba nastavak je vježbe 7.3.

Sve izmjene unesene u obrazac iz prethodne vježbe spremat će se u bazu podataka putem skripte *SpremiIzmjenu.php*.

1. U programu Brackets otvorite datoteku *SpremiIzmjenu.php (mapa „…\D351\vjezbe\vjezba7.3“.*

2. Unutar oznake za PHP kôd dodajte sljedeće naredbe:

```
 echo "Spol: $spol<br/>";
 echo "<br/>";
 $sql = "UPDATE polaznik SET Ime = '$ime', Adresa = '$adresa',
Email='$email', Spol='$spol' where Id=$id";

 if (mysqli_query($veza,$sql))
   echo "Izmjena je uspješno spremljena.";
 else
   echo "Izmjena podataka u bazi nije uspjela.";

 mysqli_close($veza);

?>
```

Naredbom UPDATE treba izmijeniti postojeće podatke u bazi podataka. Za odgovarajući redak polja će biti izmijenjena u vrijednosti koje su primljene iz obrasca.

Zatim se provjerava veza s bazom podataka te se izvršava upit pomoću funkcije *mysqli_query*. U slučaju uspješnog ili neuspješnog izvršavanja upita ispisuje se odgovarajuća poruka. Nakon toga zatvara se veza s bazom podataka.

3. Nakon bloka PHP kôda dodajte poveznicu za povratak na pregled adresa.

```
    ?>
    <br /><br />
    <a href="PregledAdresa.php">Pregled adresa</a>
</body>
</html>
```

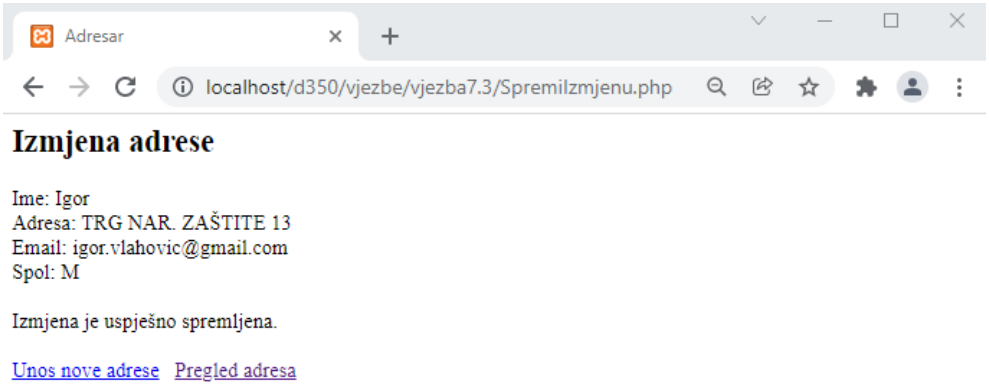
Spremite datoteku u mapu „…\D351\vjezbe\vjezba7.3“.

4. Pokrenite servis XAMPP ako već nije pokrenut.

5. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba7.3/PregledAdresa.php.

6. Pritisnite na ime polaznika.

7. Izmijenite prikazane podatke i pritisnite *Spremi*.



8. Nakon što se prikaže poruka da je izmjena spremljena, pritisnite na poveznicu *Pregled adresa*. Uočavate da su na pregledu adresa prikazani izmijenjeni podaci.

9. Pokrenite aplikaciju phpMyAdmin. (U *web*-preglednik upišite adresu: *http://localhost/*phpmyadmin/.) Uvjerite se da su podaci zaista promijenjeni u bazi.

Vježba 7.5 – Upis podatka u bazu podataka

Napomena

Ova vježba je nastavak vježbe 7.3.

U ovoj vježbi mijenja se skripta *SpremiAdresu.php* tako da se novouneseni kontakt sprema u bazu podataka.

1. U programu Brackets otvorite datoteku *SpremiAdresu.php* u mapi „…\D351\vjezbe\vjezba7.3“.

2. Dodajte sintaksu za unos podataka u bazu:

```
  echo "Email: $email<br/>";

  $sql = "INSERT INTO polaznik";
  $sql .= "(Ime, Prezime, Adresa, GradId, Spol, Email) ";
  $sql .= "VALUES ( '$ime', '$prezime',
'$adresa','$grad','$spol','$email')";

  if (mysqli_query($veza,$sql))
    echo "Podaci su uspješno spremljeni.";
  else
    echo "Spremanje podataka u bazu nije uspjelo.";

  mysqli_close($veza);
```

Najprije se unosi upit INSERT pomoću kojega će kontakt biti ubačen u bazu podataka. Primjećujete da u njemu nije potrebno navoditi vrijednost polja *Id* (za koje je u bazi podataka podešeno da se automatski povećava za 1, pomoću opcije *auto_increment*).

Nakon toga se provjerava veza s bazom. Zatim se izvršava upit te se prikazuje poruka o uspjehu upita. Na kraju se veza s bazom zatvara.

3. Spremite datoteku.

4. Pokrenite servis XAMPP ako već nije pokrenut.

5. U *web*-preglednik upišite adresu: *http://localhost/vjezbe/vjezba7.3/UnosAdrese.htm* .

6. Unesite novog polaznika i pritisnite na *Spremi*.

7. Nakon potvrde o uspješno spremljenim podacima pritisnite na poveznicu *Pregled adresa* i uvjerite se da je novi unos prikazan u listi.

Vježba 7.6 – Brisanje podatka iz baze podataka

U ovoj vježbi dodat ćemo poveznicu za brisanje pojedinog polaznika i skriptu koja će obaviti brisanje.

Napomena

Ova vježba nastavak je vježbe 7.3.

1. U programu Brackets otvorite datoteku *PregledAdresa.php* u mapi „…\D351\vjezbePHP\vjezbe\vjezba7.3“.

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 poveznicom na skriptu za brisanje:

```
   echo "<td>$spol</td>";
   echo "<td>
        <a href='ObrisiAdresu.php?Id=$id'>Obriši</a></td>";
```

```
    echo "</tr>";
```

Skripti *ObrisiAdresu.php* bit će, putem URL-a, proslijeđen *Id* polaznika kojeg treba obrisati.

(Napomena: naredbu treba napisati u jednom retku.)

Spremite datoteku.

4. U programu Brackets otvorite datoteku *ObrisiAdresu.php (u mapi „…\D351\vjezbePHP\vjezbe\vjezba7.3“)*.

5. Nakon oznake *h2* ubacite oznake za PHP kôd i sljedeće naredbe unutar njih:

```
<?php
  include("otvoriVezu.php");
  $id = $_GET['Id'];

  if (mysqli_query($veza,"DELETE FROM polaznik WHERE Id =
$id"))
  {
    echo "Zapis je obrisan.";
  }
  mysqli_close($veza);
?>
```

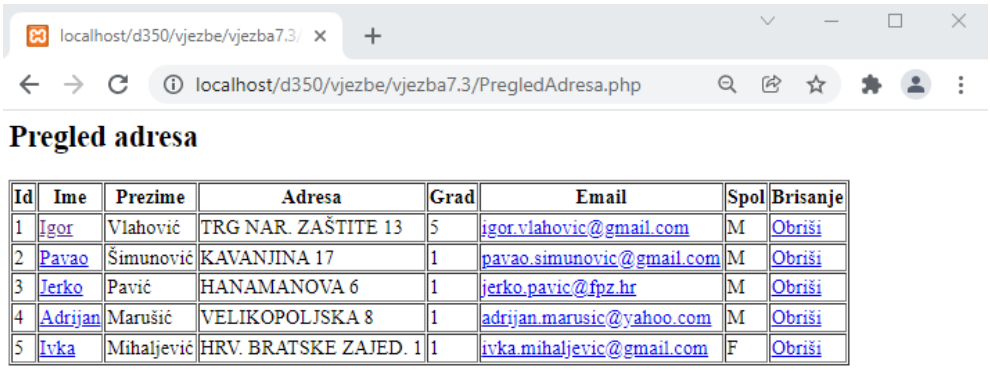
Prva naredba provjerava otvaranje veze. Zatim se iz polja *$_GET* čita *Id* kontakta, proslijeđen putem URL-a sa stranice *PregledAdresa.php*, i sprema u varijablu *$id*.

Sljedeća naredba otvara vezu s bazom podataka, a nakon toga izvršava se upit koji će obrisati redak sa zadanim *Id*jem . Ako je brisanje uspjelo, ispisat će se poruka o tome. Posljednja naredba zatvorit će vezu s bazom podataka.

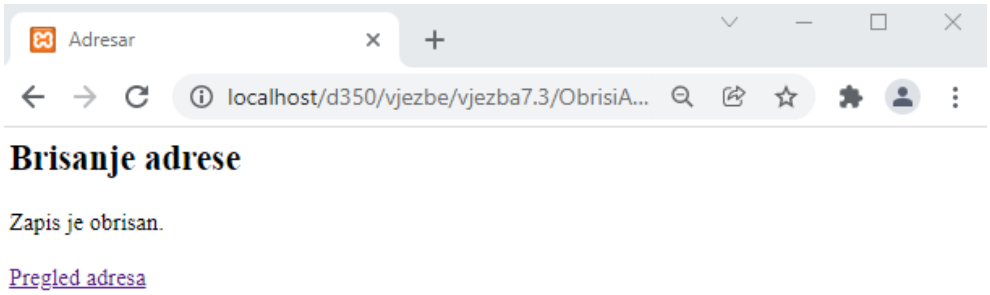
6. Spremite datoteku u mapu „…\D351\vjezbePHP\vjezbe\vjezba7.3“.

7. Pokrenite servis XAMPP ako nije već pokrenut.

8. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba7.3/PregledAdresa.php.



9. Pritisnite na jednu od poveznica za brisanje.



10. Nakon što se prikaže poruka da je brisanje uspješno obavljeno, vratite se na pregled adresa i uočite da obrisanog kontakta više nema u popisu.

8. Niz

3. dan

Trajanje poglavlja:
45 min

Po završetku ovoga poglavlja moći ćete:

- definirati koncept niza
- opisati niz s brojčanim indeksima
- objasniti asocijativni niz
- prepoznati dvodimenzionalni i višedimenzionalni niz.

Dosad smo se već upoznali s varijablama kojima možemo pridružiti određene vrijednosti. Za korištenje više vrijednosti odjednom programski jezici nude koncept niza. Niz je varijabla koja može sadržavati više vrijednosti odjednom, a svakoj vrijednosti (članu niza) pridružen je ključ. Pojedinom članu niza pristupa se pomoću ključa, koji može biti broj ili više znakova.

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

Nizovi mogu biti jednodimenzionalni, ali i dvodimenzionalni i višedimenzionalni.

8.1. Niz s brojčanim ključem (indeksom)

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

Tablica 4.1 prikazuje primjer niza koji sadrži imena gradova.

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

Za stvaranje niza koristi se ključna riječ *array*:

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

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

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

```
Rijeka
```

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

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

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

Već stvoren niz 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.

8.2. Niz sa znakovnim ključem

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

Za niz koji treba sadržavati poštanske brojeve gradova ključ niza može biti naziv grada.(Tablica 4.2)

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

Pri stvaranju niza 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 niza koristi se njegov ključ:

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

```
51000
```

Pridruživanje vrijednosti članu niza također se obavlja preko njegova ključa:

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

8.3. Dvodimenzionalni niz

Dok se jednodimenzionalni niz može predstaviti kao niz vrijednosti, dvodimenzionalni niz može se predstaviti kao tablica. Član dvodimenzionalnog niza određen je dvama ključevima – po jednim za svaku dimenziju.

Primjer dvodimenzionalnog niza moglo bi biti trenutno stanje u igri Križić-kružić. Budući da je riječ o nizu dimenzija 3 x 3, vrijednosti obaju indeksa (okomitog i vodoravnog) su od 0 do 2.

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

Tablica 4.2: Niz u koje se sprema stanje u igri Križić-kružić (u zaglavlju i u prvom stupcu su indeksi niza)

Dvodimenzionalni niz u koji se sprema gornje stanje u igri stvorit ćemo ovako:

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

Dvodimenzionalni niz stvara se tako da se definira niz koji za članove ima jednodimenzionalan niz.

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

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

```
O
```

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

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

Ako se retke, umjesto brojevima 0,1 i 2, želi označiti slovima A, B i C, potrebno je stvoriti niz 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";
  echo $igra["A"][2];
?>
```

Moguće je definirati niz neograničenog broja dimenzija, iako nizovi s više od 3 dimenzije nemaju praktičnu primjenu. Naredba kojom se stvara trodimenzionalni niz slična je naredbi za stvaranje dvodimenzionalnog niza, samo se sada definira polje čiji je svaki član dvodimenzionalni niz. Za pristup članu trodimenzionalnog niza bit će potrebna tri indeksa.

8.4. Petlja *foreach*

Petlje se često koriste za ispis članova niza (nizovi su detaljnije opisani u poglavlju 8). Kod nizova s brojčanim ključem članovi niza mogu se ispisati pomoću petlje *for*, *while* ili do...*while*. Primjer ispisa članova niza 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 niza, tako da se u svakom krugu petlje ispiše po jedan član niza.

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

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

Ova petlja obavit će naredbe unutar tijela petlje jedanput za svakog člana niza *$polje*, a vrijednost trenutnog člana niza bit će u varijabli *$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 bi se način 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 niza. Razlog tome je što se u svakom krugu petlje stvara kopija člana niza i smješta u varijablu *$broj*. Izmjena te varijable neće se odraziti na vrijednost člana niza.

8.5. Ugnježđivanje petlji

Kao i uvjetne strukture, i petlje se mogu ugnježđivati jedna u drugu. Unutrašnja petlja može se promatrati kao zaseban blok kôda. Sljedeći primjer pokazuje da 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 dviju ugniježđenih petlji 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 dviju ugniježđenih petlji *foreach*:

```
<?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
```

Vježba 8.1 – Nizovi

1. Pokrenite program Brackets i pokrenite stvaranje PHP datoteke.

2. U datoteku upišite oznake za PHP kôd i unutar njih sljedeću naredbu koja će stvoriti niz *BrPoste*, koji će služiti za spremanje brojeva pošti:

```
<?php
  $BrPoste = array(20000,32000,52100);
?>
```

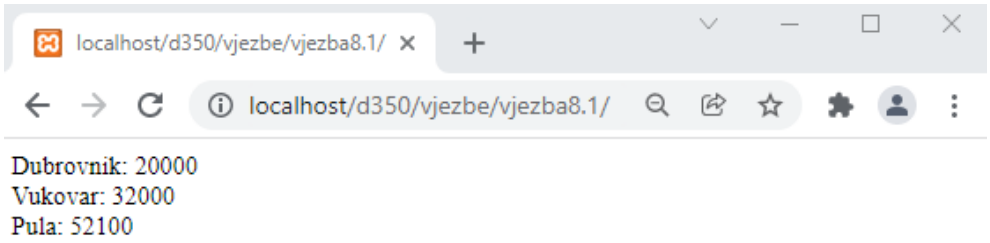
3. U datoteku upišite sljedeće naredbe, koje će ispisati pojedine gradove i pripadajuće brojeve pošti:

```
 $BrPoste = array(10000,21000,51000);

  echo "Dubrovnik: $BrPoste[0]<br />";
  echo "Vukovar: $BrPoste[1]<br />";
  echo "Pula: $BrPoste[2]";
?>
```

   Datoteku spremite u mapu „…\D351\vjezbe\vjezba8.1“ pod nazivom *index.php*.

4. Pokrenite servis XAMPP ako nije već pokrenut.

5. U web-preglednik upišite adresu: http://localhost/vjezbe/vjezba8.1/



Vježba 8.2 – Ispis niza s podacima iz tablice

1. U programu Brackets stvorite datoteku *index.php* u mapi „…\D351\vjezbePHP\vjezbe\vjezba8.2“.

2. Dodajte naredbu koja stvara niz tako da stvoreni niz ima znakovni ključ koji će biti naziv pošte.

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

   $rezultat = mysqli_query($veza,"SELECT Naziv,PostanskiBroj
   FROM grad limit 5");

   while ($redak = mysqli_fetch_array($rezultat))
   {
    $Naziv = $redak['Naziv'];
    $BrojPoste = $redak['PostanskiBroj'];
    $nizPosta = array($Naziv, $BrojPoste);
```

3. Naredbe koje ispisuju rezultat također je potrebno promijeniti:

Napomena

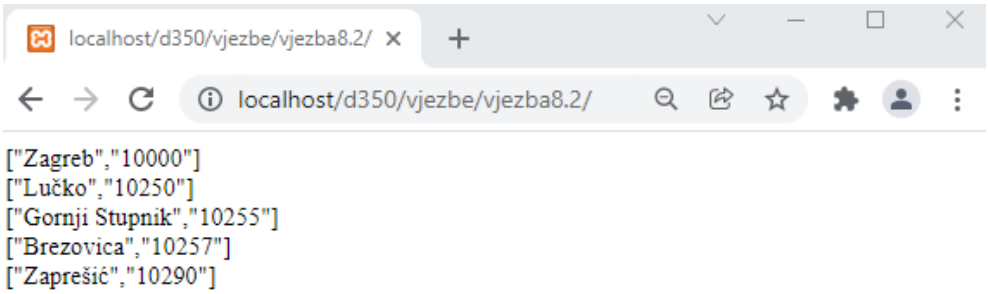
Ova vježba jedan je od primjera kako možemo koristiti niz za dohvat podataka iz tablice i ispisati rezultate niza.

```
  print json_encode($nizPosta, JSON_UNESCAPED_UNICODE);
    echo "<br>";
}
mysql_close($veza);
?>
```

Spremite datoteku.

4. Pokrenite servis XAMPP ako već nije pokrenut.

5. U *web*-preglednik upišite adresu: http://localhost/vjezbe/vjezba8.2/



Ispisat će se naziv i pripadajući brojevi prvih 5 pošta iz niza kojem su pridruženi podaci iz tablice.

Dodatna vježba – Dvodimenzionalno polje

1. U programu Brackets 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>
```

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:

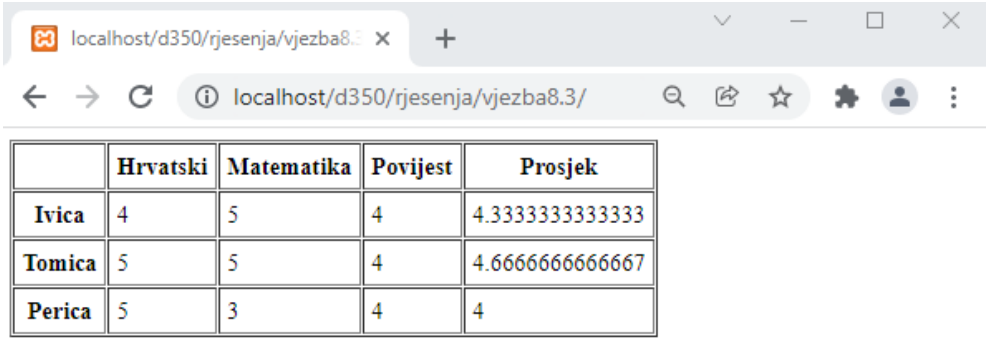
```
   </tr>
   <tr>
     <th>Ivica</th>
     <td><?php echo $ocjene["Ivica"]["Hrvatski"] ?></td>
     <td><?php echo $ocjene["Ivica"]["Matematika"] ?></td>
     <td><?php echo $ocjene["Ivica"]["Povijest"] ?></td>
     <td><?php echo $prosjekIvica ?></td>
   </tr>
   <tr>
     <th>Tomica</th>
     <td><?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 „…\D351\vjezbePHP\vjezbe\vjezba8.3“.

6. Pokrenite servis XAMPP ako već nije pokrenut.

7. U *web*-preglednik upišite adresu: http://localhost/vjezbe/vjezba8.3/ .

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

9. Funkcije

3. dan

Trajanje poglavlja:
45 min

Po završetku ovoga poglavlja moći ćete:

- definirati funkcije
- objasniti pozivanje funkcije
- opisati funkcije s argumentima i prijenos vrijednosti u funkciju
- opisati funkcije koje vraćaju rezultat
- prikazati uključivanje vanjskih datoteka u kôd
- objasniti doseg varijabli.

Ako bismo željeli pojedine naredbe ili set naredbi koristiti nekoliko puta unutar aplikacije ili pojedine skripte a da ih ne moramo svaki put pisati, možemo ih napisati jednom unutar funkcije i pozvati kada nam zatreba. Osim što time postižemo uštedu vremena, smanjujemo i mogućnost pogreške i jednostavnije je kasnije mijenjanje kôda jer se mijenja samo na jednom mjestu.

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

9.1. Funkcije

Funkcije se pišu tako da se navede ključna riječ *function*, a nakon nje 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 jer će stvoriti pogrešku pri pozivanju skripte.

Napomena

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. Preporučuje 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!";
  }
?>
```

Napomena

Za potrebe ovih primjera koristi se skripta iz mape "...D351\primjeri\poglavlje9\primjer1.php".

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 definicija funkcije u datoteci bude napisana prije njena poziva.

9.2. Funkcije s argumentima

Prilikom poziva funkciji se mogu predati određeni podaci koje će ona upotrijebiti. Ti podaci nazivaju se **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
```

Napomena

Prilikom definicije funkcije predefinirani argumenti navode se tek nakon što se navedu svi „obični” argumenti.

Predefinirani argumenti

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

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

```
<?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 predani broj množi s 2 ako se ne navede drugi argument.

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.

9.3. Ispis rezultata funkcije

Ispis rezultata funkcije može se ostvariti 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*.

9.4. 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);
  echo $c;
?>
```

```
25
```

Napomena

Kod rekurzivne funkcije bitno je postojanje tzv. početnog slučaja. Ako je on ispunjen, funkcija ne poziva opet samu sebe, već vraća rezultat.

Funkcija može pozvati samu sebe. Takva funkcija naziva se **rekurzivnom funkcijom**. Klasičan primjer takve funkcije je ispis brojeva od 1 do 10 .

```
<?php
  function Brojevi($number)
      {
            if($number<=10)
            {
                  echo "$number ";
                  Brojevi($number+1);
            }
      }

  Brojevi(1);
?>
```

```
1 2 3 4 5 6 7 8 9 10
```

9.5. Uključivanje vanjskih datoteka u kôd

Ako je potrebno pozvati funkciju unutar neke druge skripte u aplikaciji, nije potrebno ponovno pisati funkciju, već se u novoj skripti samo pozove datoteka u kojoj je funkcija napisana.

Funkcija za množenje dvaju brojeva 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);
  echo $c;
?>
```

Datoteka se može uključiti i upotrebom naredbe *require*:

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

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

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 tom bi se slučaju prekinulo i izvršavanje skripte u koju je datoteka uključena.

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*.

Napomena

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

9.6. Doseg varijabli

Varijable smo već obrađivali u prethodnim poglavljima i poznato nam je njihovo djelovanje unutar skripti. Upotreba varijabli posebno dolazi do izražaja kada se koriste funkcije, tj. kada se pozivaju funkcije koje se nalaze u vanjskim skriptama, a poziva ih se unutar postojeće skripte.

Važno je napomenuti da se varijable mogu koristiti ne samo između pojedinih skripti, već je moguće koristiti varijable i unutar svih aplikacija na poslužitelju, tj. unutar cijelog poslužitelja.

Varijable se po svom dosegu mogu podijeliti na tri vrste:

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

Lokalne varijable

Lokalne varijable vidljive su samo unutar funkcije ili skripte 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 pripadaju i argumenti funkcije, koji neće imati veze s eventualnim istoimenim varijablama u ostatku skripte.

Globalne varijable

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

Ako se neka varijabla želi koristiti kao globalna varijabla, potrebno ju je definirati pomoću ključne riječi *global*.

Napomena

U PHP-u nema deklariranja varijable (navođenja tipa podataka koji varijabla koristi).

```
<?php
  function Ispisi()
  {
    global $a;
    echo $a;
  }
  $a = 1;
  Ispisi();
?>
```

```
1
```

Predefinirane globalne varijable

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

U polju *$GLOBALS*, koje je dostupno u svim skriptama, nalazit će se globalne varijable definirane u trenutnoj skripti. Pojedinom članu tog polja pristupa se putem 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. Riječ je o poljima koja sadrže vrijednosti vezane za postavke poslužitelja, HTTP zahtjeva i slično. Ova polja također imaju znakovne ključeve putem 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
```

Napomena

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 prenesene u HTTP zahtjevu metodom *POST*
$_COOKIE	vrijednosti zapisane u kolačićima (tekstualnim datotekama koje *web*-preglednik sprema na korisnikovu 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 9.1 – Funkcija

1. Pokrenite program Brackets i pokrenite stvaranje PHP datoteke.

2. U datoteci napišite funkciju koja računa prosjek triju brojeva i vraća izračunati rezultat:

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

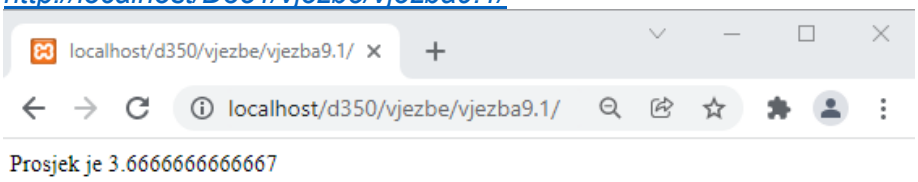
3. Stvorite tri varijable, pridružite im vrijednosti i izračunajte njihov prosjek:

```
  }

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

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

   Spremite datoteku u mapu „…\D351\vjezbe\vjezba9.1“ pod nazivom *index.php* .

4. Pokrenite servis XAMPP ako već nije pokrenut.

5. U *web*-preglednik upišite adresu:
   http://localhost/D351/vjezbe/vjezba9.1/



Vježba 9.2 – Uporaba funkcije

Napomena

Ova vježba obavlja istu radnju kao i prethodna, samo što se ovdje koristi pozivanje funkcije iz vanjske skripte.

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

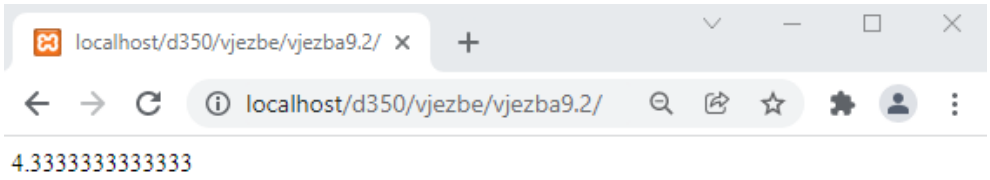
```
<?php
  function Prosjek($a, $b, $c)
  {
    return ($a + $b + $c) / 3;
  }
?>
```

   Spremite datoteku u mapu „…\D351\vjezbe\vjezba9.2“ .

3. U programu Brackets stvorite datoteku *index.php* u mapi „…\D351\vjezbe\vjezba9.2“.
4. Dodajte naredbu koja će uključiti datoteku *prosjek.php* u kôd i zatim naredbu koja će računati prosjek vrijednosti pomoću funkcije *Prosjek*:

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

$prosjek = Prosjek(4,5,4);

echo $prosjek;
```

5. Spremite datoteku.
6. Pokrenite servis XAMPP ako već nije pokrenut.
7. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba9.2/



Dobiveni rezultat isti je kao i u vježbi 9.1, ali je ovdje prosjek izračunat pomoću funkcije koja računa prosjek triju brojeva.

10. Neke ugrađene funkcije PHP-a

4. dan

Trajanje poglavlja:
90 min

Po završetku ovoga poglavlja moći ćete:

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

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 dan je kratak pregled samo nekih od njih. Za primjer ćemo koristiti bazu *tecajevi*.

10.1. Funkcije za rad sa znakovnim nizovima

Funkcija ***trim*** koristi se za uklanjanje praznina s početka i kraja znakovnog niza. Pod prazninama se smatraju razmaci, tabulatori i znakovi za novi red. Funkcija vraća niz iz kojeg su uklonjene praznine ili zadani znakovi. Primjer korištenja funkcije *trim* (…D351\primjeri\poglavlje10\primjer1.php):

```
   while ($redak = mysqli_fetch_array($rezultat))
     {
         $adresa = $redak['Adresa'];
         $adresa = "\t $adresa \n";
         $adresa = trim($adresa);
         echo $adresa."<br/>";
     }
```

```
TRG NAR. ZAŠTITE 13
KAVANJINA 17
HANAMANOVA 6
```

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

```
   while ($redak = mysqli_fetch_array($rezultat))
     {
         $ime = $redak['Ime'];
         $malaslova = mb_strtolower($ime);
         $velikaslova = mb_strtoupper($ime);
         echo $velikaslova."<br/>";
     }
```

```
IGOR
PAVAO
JERKO
```

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

```
   $ime = $redak['Ime'];
   $malaslova = mb_strtolower($ime);
   $velikaslova = mb_strtoupper($ime);
   $duljina = strlen($ime);
   echo $velikaslova.' '.$duljina."<br/>";
```

```
IGOR  4
PAVAO 5
JERKO 5
```

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 (opcijski). Funkcija vraća dobiveni podniz.

Primjer njena korištenja:

```
   $adresa = $redak['Adresa'];
   echo $adresa."<br/>";
   $adresa = substr($adresa,4);
   echo $adresa."<br/>";
   $adresa = substr($adresa,4,3);
   echo $adresa."<br/>";
```

```
TRG NAR. ZAŠTITE 13
NAR. ZAŠTITE 13
ZA
KAVANJINA 17
NJINA 17
A 1
HANAMANOVA 6
MANOVA 6
VA
```

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 (opcijski) argument može se predati broj obavljenih zamjena. Funkcija vraća niz nastao zamjenom. Primjer korištenja ove funkcije:

```
   $ime = $redak['Ime'];
   $ime = str_replace('a','xxx',$ime);
   echo $ime."<br/>";
```

```
Igor
Pxxxvxxxo
Jerko
```

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

```
   $adresa = $redak['Adresa'];
```

```
  $adresa = explode(" ",$adresa);
  foreach ($adresa as $clan)
    {
       echo $clan . "<br />";
    }
```

```
TRG
NAR.
ZAŠTITE
13
KAVANJINA
17
HANAMANOVA
6
```

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 niza u znakovni niz:

```
  $adresa = $redak['Adresa'];
  $adresa = explode(" ",$adresa);
  $adresa= ''.implode(' ', $adresa).'';
          print_r($adresa);
          echo "<br/>";
```

```
TRG NAR. ZAŠTITE 13
KAVANJINA 17
HANAMANOVA 6
```

10.2. 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*** nasumično će 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 sortira se 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 njegova 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).

U sljedećem primjeru za prikaz ključeva i vrijednosti koristit ćemo petlju *foreach.* Petlja vraća polje koje sadrži ključ i vrijednost trenutnog člana koji se dodjeljuju varijablama „name“ i „value“. Primjer ispisivanja članova pomoću petlje *while*:

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

foreach ($names as $name => $value) {
    echo $name." ".$value."</br>";
}
?>
```

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

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

Napomena

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

10.3. Funkcije za rad s datumima i vremenom

U jeziku PHP za rad s datumima i vremenima koristi se format poznat kao Unixova *vremenska oznaka*. Riječ je 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 opcijski, a ako se izostave, koriste se trenutne vrijednosti: trenutna godina, trenutni dan, trenutni mjesec itd.

Napomena

Opcijski argumenti se prilikom pozivanja funkcije *mktime* (kao i kod ostalih funkcija s opcijskim 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 03. 02. 2022. i vrijeme 15:20:05:

```
<?php
  $datum = mktime(15,20,5,02,03,2022);
  echo $datum;
?>
```

```
1643898005
```

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)

Primjer korištenja funkcije *date*:

```
$DatumRodjenja = $redak['DatumRodjenja'];
$DatumRodjenja = date("d.m.Y.",
strtotime($DatumRodjenja));
echo $DatumRodjenja."<br/>";
```

```
25.02.1993.
30.07.1997.
07.06.1994.
```

Funkcija ***getdate*** koristi se za dohvaćanje pojedinog podatka iz vremenske oznake. Uzima vremensku oznaku kao argument, a ako se pozove bez argumenta, uzima se trenutno vrijeme. ***Getdate*** može koristiti atribute:

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*:

```
  $datum = mktime(15,20,5,02,03,2022);
  $vrijeme = getdate($datum);

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

```
3.2.2022 15:20:5
```

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(13,1,2022))
  {
    echo "Datum nije ispravan!";
  }
?>
```

```
Datum nije ispravan!
```

U gornjem primjeru funkcija *checkdate* vraća vrijednost *FALSE* jer datum 1. 13. 2022. nije ispravan. Ovu funkciju dobro je koristiti za provjeru datuma upisanog od strane korisnika.

10.4. Matematičke funkcije

Funkcija ***round*** koristi se za zaokruživanje decimalnog broja. Argumenti koji se predaju funkciji su broj koji treba zaokružiti i (opcijski) 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.

Napomena

Funkcija *round* može imati i treći argument *mode*, koji može imati jednu od 4 vrijednosti: PHP_ROUND_HALF_UP, PHP_ROUND_HALF_DOWN, PHP_ROUND_HALF_EVEN i PHP_ROUND_HALF_ODD.

Ove vrijednosti mogu zaokruživati i po pola broja.

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* (…d250\primjeri\poglavlje10\primjer4.php):

```
$prosjek = $redak['Prosjek'];
echo $prosjek."<br/>";
$prosjekR = round($prosjek,2);
echo $prosjekR."<br/>";
$prosjekC = ceil($prosjek);
echo $prosjekC."<br/>";
$prosjekF = floor($prosjek);
echo $prosjekF."<br/>";
```

```
409.617308
409.62
410
409
```

Agregatne funkcije mogu se koristiti u nizovima ili u samom SQL upitu na tablicu. Ovdje ćemo koristiti agregatne funkcije postavljene u SQL upitu na tablicu *tecaj* (*SELECT Cijena, Popust, count(*) as Broj, min(Cijena)as Min, max(Cijena)as Max FROM tecaj*).

Funkcija ***max*** vraća najveću cijenu tečaja, a funkcija ***min*** vraća cijenu tečajeva.

```
      $min = $redak['Min'];
      $max = $redak['Max'];

      echo "Min:".$min."<br/>";
      echo "Max:".$max."<br/>";
```

```
Min:0.00
Max:750.00
```

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 16:

```
<?php
  echo sqrt(16);
?>
```

```
4
```

Napomena

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

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

```
<?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.

10.5. 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) 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 uvijek će 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*.

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

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

Vježba 10.1 – Prikaz datuma i vremena

1. Pokrenite program Brackets i pokrenite stvaranje PHP datoteke.

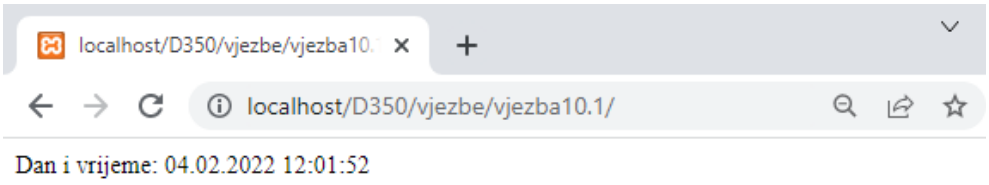
2. U datoteci spremite trenutno vrijeme u varijablu i ispišite ga:

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

3. Spremite datoteku u mapu „...\D351\vjezbe\vjezba10.1“ pod nazivom *index.php*.

4. Pokrenite servis XAMPP ako već nije pokrenut.

5. U *web*-preglednik upišite adresu:
   http://localhost/D351/vjezbe/vjezba10.1/ .

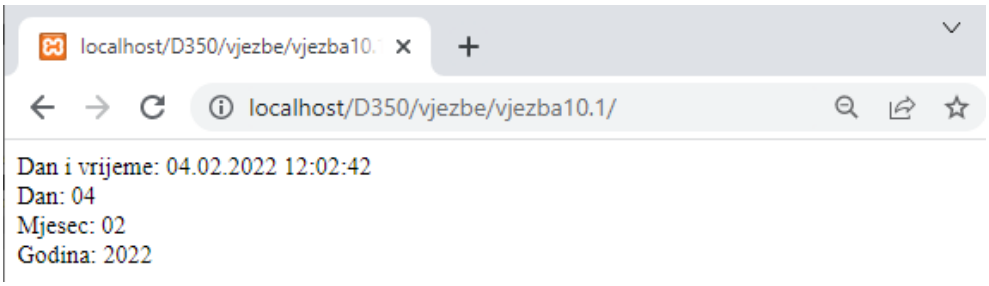


U sljedećim koracima vrijednost trenutnog datuma podijelit ćemo na dijelove.

6. Podijelit ćemo vr*ijeme* na dan, mjesec i godinu:

```
$Dan = date("d");
$Mjesec = date("m");
$Godina = date("Y");
echo "Dan: ".$Dan."<br/>";
echo "Mjesec: ".$Mjesec."<br/>";
echo "Godina: ".$Godina."<br/>";
```

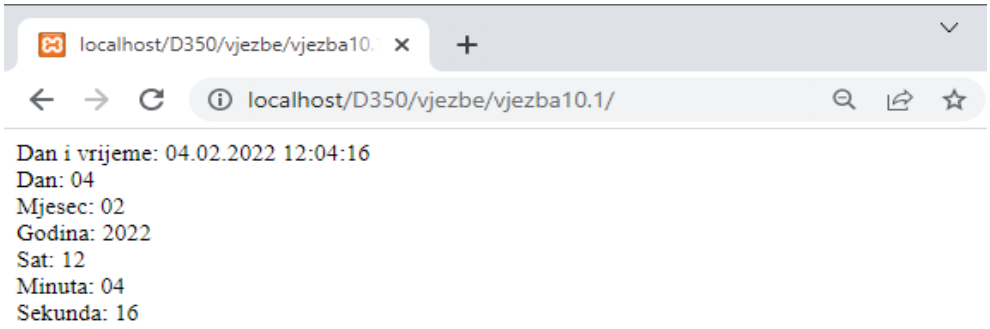
Spremite datoteku i osvježite stranicu u *web*-pregledniku.



7. Podijelit ćemo v*rijeme* još i na sat, minute i sekunde:

```
$sat = date("H");
$minuta = date("i");
$sekunda = date("s");

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

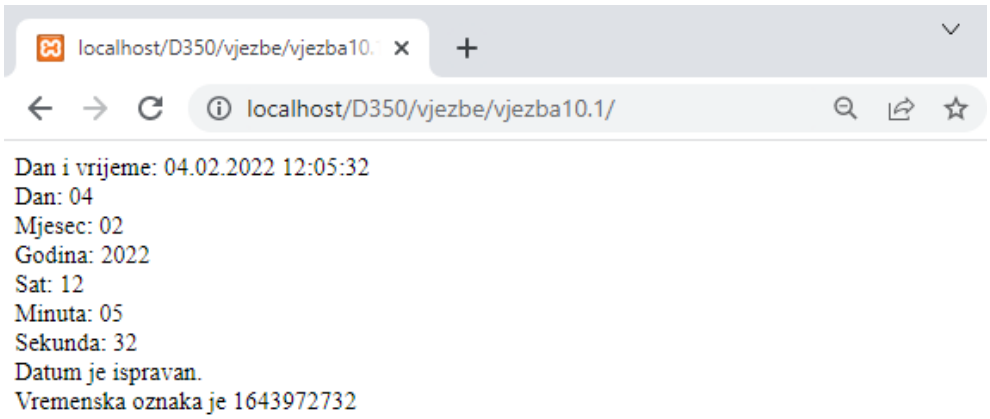
8. Spremite datoteku i osvježite stranicu u *web*-pregledniku.



9. Pomoću funkcije *checkdate* provjerite je li dobiveni datum ispravan. 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);
}
?>
```

10. Spremite datoteku i osvježite stranicu u *web*-pregledniku.



Napomena

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

Vježba 10.2 – Zaokruživanje prosjeka

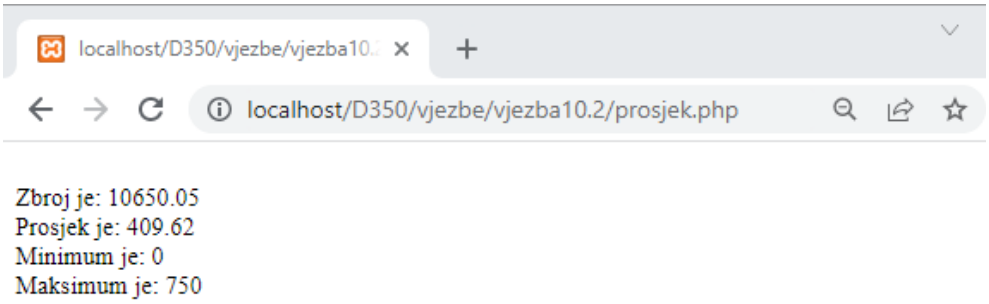
1. Pokrenite program Brackets i otvorite datoteku *prosjek.php* u mapi „…\D351\vjezbe\vjezba10.2\“.

2. Dodajte u skriptu naredbe koje će prosjek zaokružiti na dvije decimale i prikazati ukupan zbroj i prosjek cijene tečaja iz tablice *tecaj*. Također je

potrebno dodati naredbe koje će prikazati i minimalnu i maksimalnu cijenu tečaja:

```
  $rezultat = mysqli_query($veza,"SELECT Cijena, Popust,
count(*) as Broj, sum(Cijena) as Zbroj, avg(Cijena) as Prosjek,
min(Cijena)as Min, max(Cijena)as Max FROM tecaj");
    while ($redak = mysqli_fetch_array($rezultat))
    {
      $cijena = $redak['Cijena'];
      $popust = $redak['Popust'];
      $broj = $redak['Broj'];
      $prosjek = $redak['Prosjek'];
      $zbroj = $redak['Zbroj'];
      $min = $redak['Min'];
      $max = $redak['Max'];

      echo "Zbroj je: ".round($zbroj,2)."<br/>";
      echo "Prosjek je: ".round($prosjek,2)."<br/>";
      echo "Minimum je: ".round($min,2)."<br/>";
      echo "Maksimum je: ".round($max,2)."<br/>";
    }
    mysqli_free_result($rezultat);
```

3. Spremite datoteku.
4. Pokrenite servis XAMPP ako već nije pokrenut.
5. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba10.2/prosjek.php



Izračunati prosjeci prikazani su tako da su zaokruženi na dvije decimale

11. Rad s datotekama

4. dan

Trajanje poglavlja:

45 min

Po završetku ovoga poglavlja moći ćete:

- otvarati datoteke pomoću funkcije *fopen*
- zatvarati datoteke pomoću funkcije *fclose*
- opisati čitanje iz datoteke pomoću funkcije *fread*
- opisati čitanje jednog retka iz datoteke pomoću funkcije *fgets*
- opisati čitanje jednog znaka iz datoteke pomoću funkcije *fgetc*
- objasniti ispitivanje je li pročitan znak za kraj datoteke pomoću funkcije *feof*
- opisati pisanje u datoteku pomoću funkcije *fwrite*
- slati datoteke na poslužitelj pomoću elementa obrasca input s atributom *type="file"*
- spremati poslane datoteke na poslužitelj pomoću funkcije *move_uploaded_file*.

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“.

Napomena

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

11.1. Otvaranje i zatvaranje datoteke

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

Za primjer koristit ćemo datoteke *UnosAdrese.html* i *SpremiAdresu.php* koje se nalaze u mapi „…D351/primjeri/poglavlje11/”.

U programu Brackets otvorit ćemo datoteku *SpremiAdresu.php* i unutar datoteke upisati sljedeći PHP kôd:

```
$datoteka = fopen ("adresar.txt","a");
```

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

Prvi argument koji funkcija *fopen* prima je putanja do datoteke. Putanja do datoteke može biti apsolutna (potpuna putanja do datoteke, npr. „…/D351/primjeri/poglavlje11.1/datoteka.txt“ ili relativna (putanja u odnosu na mapu gdje se nalazi skripta, npr. „*Primjer11.1/datoteka.txt*“). Da bi skripte bez problema mogle raditi na različitim poslužiteljima, bolje je upotrebljavati relativne putanje.

Drugi argument funkcije je način rada s datotekom. Riječ je 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:

Napomena

Relativne putanje promatraju se 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 Windowsovim poslužiteljima za navođenje apsolutne putanje može se, ali i ne mora, koristiti i obrnuta kosa crta („\”).

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 na kraj u datoteku (postojeći sadržaj se ne briše)
a+		dodavanje na kraj 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

Preporučuje 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.

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 njena 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.

11.2. Zapisivanje u datoteku

Primjere ćemo započeti zapisivanjem podataka u datoteku.

Nakon poziva funkcije *fopen* provjerit ćemo je li datoteka otvorena jer u protivnom neće biti moguće izvršiti radnju (čitanje ili pisanje) nad datotekom.

Datoteku *SpremiAdresu.php* otvorit ćemo u programu Brackets i unijeti sljedeću sintaksu prije završetka PHP oznake:

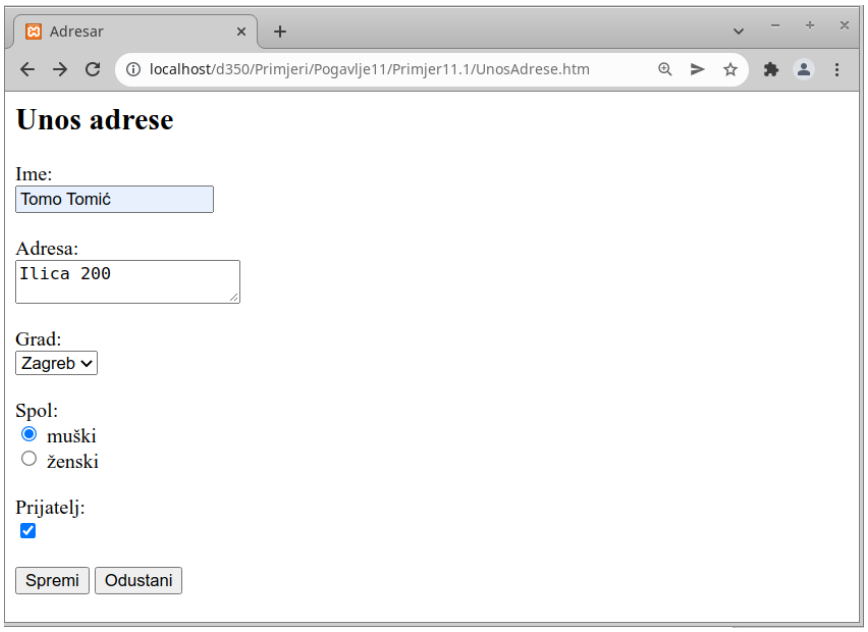
```
echo "Prijatelj: $prijatelj <br/>";
$datoteka = fopen ("adresar.txt","a");

?>
```

Za provjeru je li datoteka otvorena koristit ćemo uvjetnu strukturu i unutar nje dodati sintaksu za spremanje podataka u datoteku:

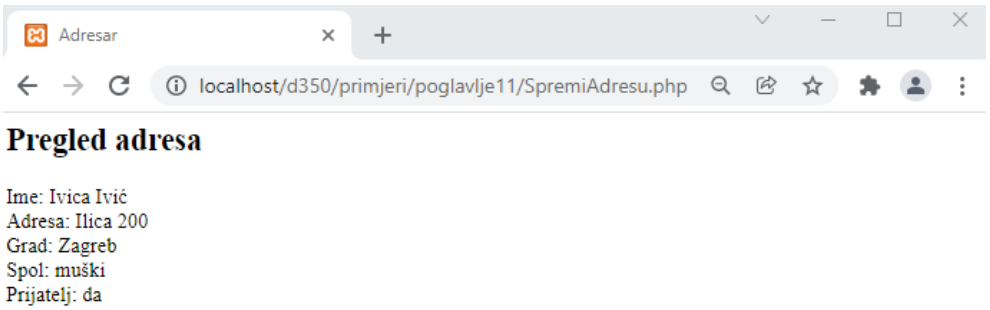
```
echo "Prijatelj: $prijatelj <br/>";
$datoteka = fopen ("adresar.txt","a");

if ($datoteka)
{
   fwrite($datoteka,
 "\n$ime\n$adresa\n$grad\n$spol\n$prijatelj");
}
?>
```

Za zapisivanje podataka u datoteku koristi se funkcija *fwrite* koja otvara datoteku i upisuje sadržaj u navodnicima u datoteku (prijelom podataka je ovdje naveden iz grafičkih razloga).

Otvorit ćemo datoteku *UnosAdrese.html* i unijeti podatke.

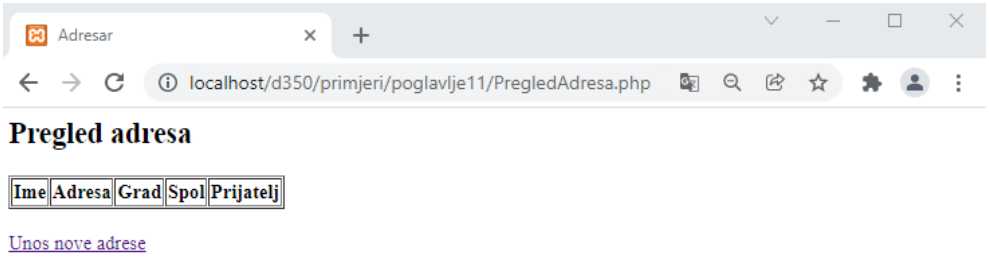


Pritiskom na *Spremi* podaci će se pohraniti u datoteku i ispisati u skripti *SpremiAdresu.php*.



11.2. Čitanje iz datoteke

Za primjer koristit ćemo datoteku *PregledAdresa.php.*



Da bismo ispisali podatke iz datoteke *Adresar.txt*, koristimo funkciju ***fgets***, koja 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.

U programu Brackets otvorit ćemo datoteku *PregledAdresa.php*.

Prvo ćemo upisati sintaksu za otvaranje datoteke:

```
<?php
$datoteka = fopen("Adresar.txt", ”r”);
```

Da ispišemo sadržaj cijele datoteke, unutar PHP oznaka upisat ćemo sintaksu koja će napraviti čitanje datoteke redak po redak:

```
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);
}
?>
```

Unutar petlje *while* izvršava se čitanje jednog retka iz datoteke. Nakon što se pročita posljednji redak, funkcija *feof* vratit će *false* i petlja će se prestati izvršavati.

Na kraju dodajemo naredbu za zatvaranje datoteke **fclose**.

11.4. Slanje datoteke na poslužitelj

Datoteka se može poslati s korisnikova 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 mora se koristiti metoda POST.

```
<form method="POST" action="SpremiAdresu.php"
 enctype="multipart/form-data">

   <input type="file" name="datoteka"/>

</form>
```

Na poslužiteljskoj strani, poslanoj datoteci i informacijama o njoj može se pristupiti putem polja *$_FILES,* slično kao što se drugim podacima upisanim u obrazac može pristupiti putem polja *$_GET* odnosno *$_POST*.

Polje *$_FILES* dvodimenzionalno je polje, u kojem prva dimenzija označava naziv polja za odabir (atribut *name*) budući da ih može biti više unutar istog obrasca. Druga dimenzija odnosi se na pojedinu vrijednost vezanu uz datoteku. Polje *$_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 formatu MIME)
$_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["slika"]["tmp_name"]
{
   // spremi poslanu datoteku
}
```

Također, uputno je provjeriti je li se dogodila greška prilikom slanja, pa će potpunija provjera izgledati ovako:

```
if ($_FILES["slika"]["tmp_name"] &&
    && $_FILES["slika"]["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.

Napomena

Pritiskom na *Browse* otvara se samo dijaloški okvir za odabir datoteke na klijentskom računalu. Nakon što se datoteka odabere i pritisne OK, datoteka još nije poslana. Slanje datoteke na poslužitelj događa se tek kad se pritisne na tipku obrasca za slanje podataka (tipku tipa *submit*).

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($_FILES["slika"]["tmp_name"],
"Slike/$ime.jpg");
```

Prvi argument funkcije je trenutna, privremena putanja do datoteke, a drugi je željena putanja.

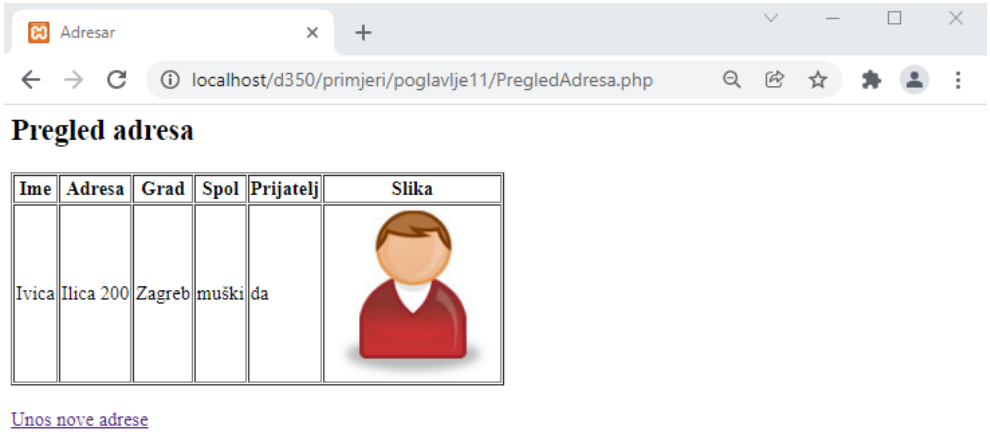
Kako bismo provjerili je li slika uspješno dodana na lokaciju, izmijenit ćemo datoteku *PregledAdresa.php* tako da dodamo stupac u tablici:

```
<th>Prijatelj</th>
  <th>Slika</th>
</tr>
```

… i u stupcu dodamo kod za prikaz slike:

```
 echo "<td>$prijatelj</td>";
 echo "<td><img src='Slike/$ime.jpg'></td>";
 echo "</tr>";
```

Pokrenemo PregledAdresa.*php* i dobijemo rezultat:



Vježba 11.1 – Slanje datoteke na poslužitelj

Napomena

Za ovu vježbu koristi se povezivanje na bazu *tecajevi*, a podaci se pohranjuju u tablicu *polaznik*.

1. Pokrenite program Brackets. U mapi *„…\D351\vjezbe\vjezba11.1“* 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.

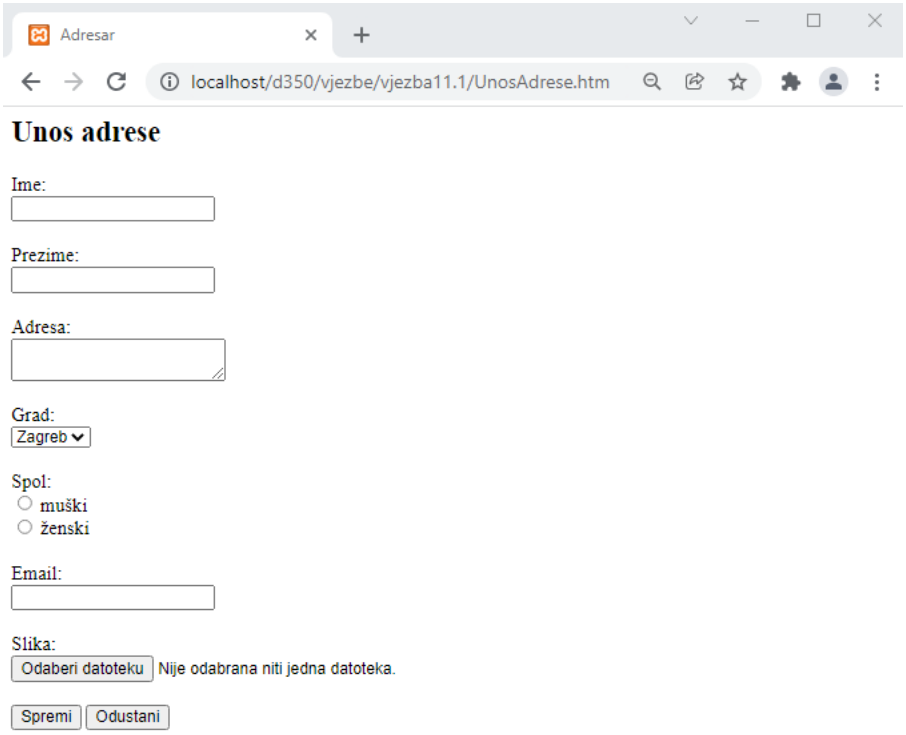
2. Prije pritiska na *Spremi* dodajte novi element u obrazac – polje za odabir datoteke, zajedno s opisnim tekstom i nekoliko oznaka *br* (iz estetskih razloga).

```
Email:<br/>
<input type="text" name="email">
<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 XAMPP (ako već nije pokrenut).

5. U *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba11.1/UnosAdrese.htm.



Obrazac za unos adrese sada će izgledati kao na slici.

6. U programu Brackets otvorite datoteku *SpremiAdresu.php* koja se nalazi u mapi *„…\D351\vjezbe\vjezba11.1*". Unutar datoteke, dodajte masno otisnuti kôd:

```
echo "Email: $email<br/>";

if ($_FILES["slika"]["name"] &&
        $_FILES["slika"]["size"] != 0)
{
   $izvor       = $_FILES["slika"]["tmp_name"];
   $odrediste   = "Slike/{$ime}.jpg
```

Provjera je li datoteka poslana na poslužitelj obavit će se ispitivanjem postoji li element *$_FILES["slika"]["name"]* u varijabli *$_FILES*. Također, postojanje datoteke može se dodatno potvrditi provjerom je li veličina poslane datoteke (element *$_FILES["slika"]["size"]*) različita od nule. Ako datoteka nije poslana, pokušaj spremanja rezultirao bi pogreškom, pa je zato potrebna provjera.

7. Unutar gornjeg bloka *if* dodajte masno otisnutu naredbu:

```
echo "Email: $email<br/>";

if ($_FILES["slika"]["name"] &&
        $_FILES["slika"]["size"] != 0)
{
   $izvor       = $_FILES["slika"]["tmp_name"];
   $odrediste   = "Slike/{$ime}.jpg";
   move_uploaded_file( $izvor, $odrediste );
}

  $slika =$odrediste;
  $sql = "INSERT INTO polaznik";
```

Varijabla *$izvor* koristi se da bi se postavilo trenutno mjesto datoteke, a varijabla *$odrediste* prikazuje mjesto gdje bi se datoteka trebala spremiti.

Funkcija *move_uploaded_file* premjestit će poslanu datoteku sa *$izvor* na novo mjesto *$destination* u mapi *Slike* (unutar trenutne mape, dakle *„…\D351\vjezbe\vjezba11.1\Slike*“). Datoteka će biti spremljena pod nazivom jednakim imenu osobe koja se upisuje u tablicu. 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 „…\D351\vjezbe\materijali\adresar“. Pritisnite *Spremi*.

10. Otvorite mapu *„…\D351\vjezbe\vjezba11.1\Slike*“. Unutar mape nalazit će se slika s nazivom jednakim imenu osobe koje je bilo uneseno u obrazac.

11. U programu Brackets otvorite datoteku *PregledAdresa.php* koja se nalazi u mapi *…\D351\vjezbe\vjezba11.1*".

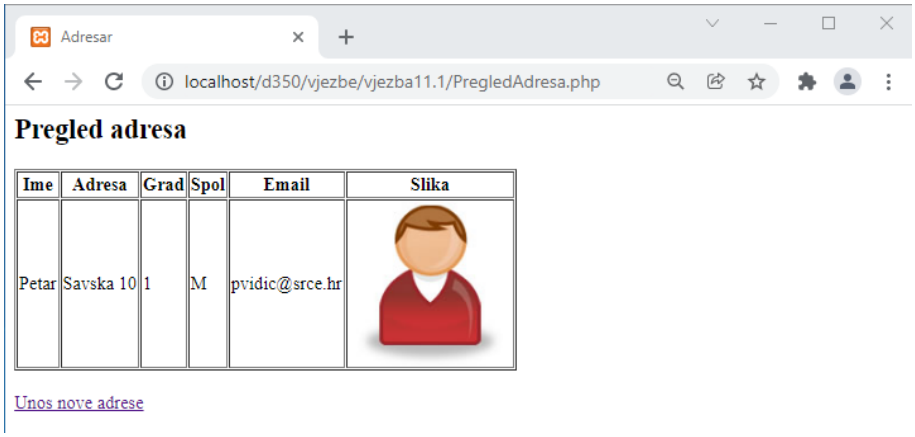
12. Među naredbama koje ispisuju pojedini redak tablice izmijenite naredbu koja prikazuje sliku:

```
echo "<td>$spol</td>";
echo "<td>$email</td>";
echo "<td><img src='$slika'></td>";
echo "</tr>";
```

Ova naredba ispisat će oznaku img pomoću koje će se prikazati slika s lokacije koja je postavljena u varijabli *$slika*.

13. Spremite datoteku.

14. U *web*-pregledniku pritisnite na poveznicu *Pregled adresa*.



Unutar tablice, zajedno s ostalim podacima, bit će prikazana i slika koja je poslana na poslužitelj putem obrasca.

12. Slanje poruke elektroničke pošte

4. dan

Trajanje poglavlja:
45 min

Po završetku ovoga poglavlja moći ćete:

- objasniti slanje poruke elektroničke pošte pomoću funkcije *mail*
- prikazati podešavanje postavki za slanje elektroničke pošte pomoću funkcije *ini_set*
- opisati slanje podataka unesenih u obrazac putem 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 tipku za slanje podataka.

12.1. 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 dostupna je samo pod operacijskim sustavom Windows)
smtp_port	port putem 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:

Napomena

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 opcijski argument u kojem se navode dodatna zaglavlja.

12.2. Slanje poruke elektroničke pošte

Za slanje poruke elektroničke pošte u jeziku PHP koristi se funkcija *mail*. Argumenti koje ona prima su:

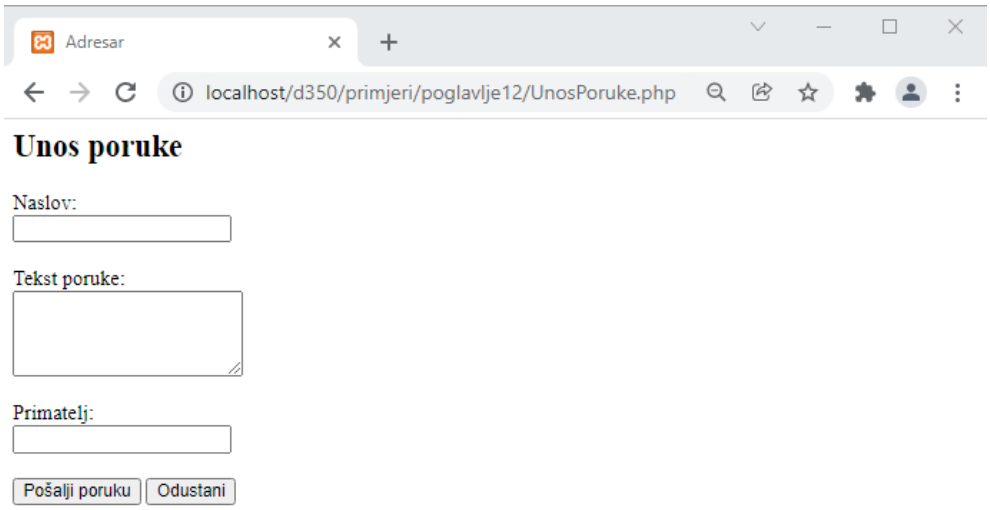
- adresa primatelja (ako je primatelja više, adrese moraju biti odvojene zarezom); adresa može biti u jednom od ovih dvaju oblika:
  - ime.prezime@domena.com
  - Ime Prezime <ime.prezime@domena.com>
- naslov poruke – u naslovu poruke ne smiju se nalaziti znakovi za novi red
- tekst poruke – kao znak za novi red u tekstu poruke treba se koristiti znak „\n“, a linija može sadržavati maksimalno 70 znakova
- dodatna zaglavlja (opcijski) – pošiljatelj poruke (*From*), dodatni primatelji (*Cc*, *Bcc*) i druga; zaglavlja moraju biti odvojena kombinacijom znakova „\r\n“
- dodatni parametri (opcijski) – služe za prosljeđivanje parametara programu na poslužitelju koji šalje poruku

Funkcija *mail* vraća vrijednost *TRUE* ako je poruka uspješno poslana, a *FALSE* ako slanje nije uspjelo.

Za unos poruke koristit ćemo obrazac *UnosPoruke.php* iz mape „*…\D351\Primjeri\Poglavlje12*“.

Napomena

Adresa pošiljatelja koja se navodi prilikom slanja poruke nigdje se neće provjeravati. Ta adresa ne mora biti postojeća, no svakako se ne preporučuje upotrebljavati nečiju tuđu adresu jer će tako poslane poruke izgledati kao da su došle od te osobe.



Za slanje poruke koristit ćemo datoteku *PosaljiPoruku.php* iz mape *„…\D351\Primjeri\Poglavlje12*“. Unutar PHP oznaka dodat ćemo sljedeću sintaksu:

```
<?php
   $email= $_POST['email'];
   $naslov = $_POST['naslov'];
   $tekst = $_POST['tekst'];

   echo "Primatelj: $email<br/>";
   echo "Naslov: $naslov<br/>";
   echo "Tekst poruke: $tekst<br/><br/>";

   ini_set("SMTP", "baltazar.srce.hr");
   ini_set("sendmail_from", "ime.prezime@srce.hr");

   if (mail($email, $naslov, $tekst))
   {
     echo "Poruka je uspješno poslana.<br/>";
   }
   else
   {
      echo "Poruka nije uspješno poslana.<br/>";
   }
?>
```

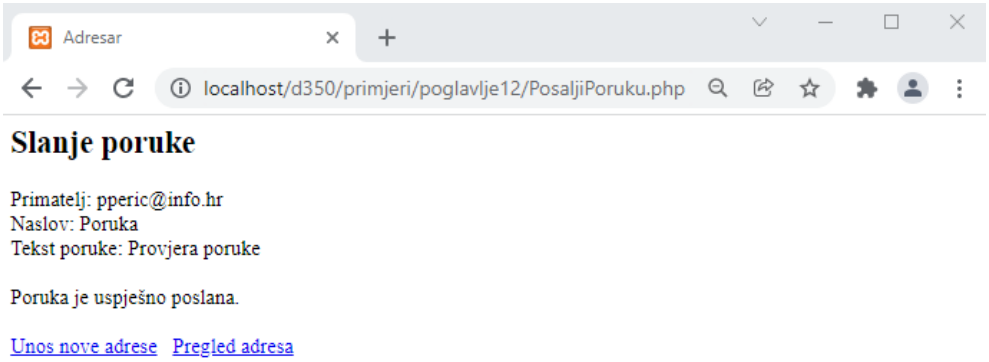
U varijabli *$email* pridružujemo vrijednost polja *Primatelj* iz obrasca, varijabli *$naslov* sadržaj polja *Naslov* poruke, a varijabli *$tekst* sadržaj poruke koju šaljemo iz obrasca.

Slijedi postavljanje postavki poslužitelja za slanje *e-mail* poruka (ovdje je to samo kao opcija, nije nužno postaviti ako je već postavljeno u datoteci *php.ini*)

U uvjetnoj proceduri provjeravaju se atributi slanja, tj. provjerava se izvođenje fukcije ***mail***. Ako su svi atributi pravilno postavljeni, javit će se poruka o uspješno poslanoj poruci. U protivnom javit će se pogreška u slanju.

Popunimo obrazac za unos poruke i pritisnemo *Pošalji*.

Ako su svi parametri dobro postavljeni, dobit ćemo obavijest da je poruka uspješno poslana.



Vježba 12.1 – Slanje podataka iz obrasca putem poruke elektroničke pošte

U tablici iz baze podataka imamo tekstualno polje u kojem se sprema adresa elektroničke pošte, a ta adresa će se, u obliku poveznice, prikazivati prilikom pregleda liste unosa. Pritiskom na poveznicu otvorit će se novi obrazac pomoću kojeg će se moći poslati poruka na navedenu adresu.

1. Pokrenite program Brackets. U mapi *„…\D351\vjezbe\vjezba12.1*“ otvorit ćemo datoteku *PosaljiPoruku.php*.

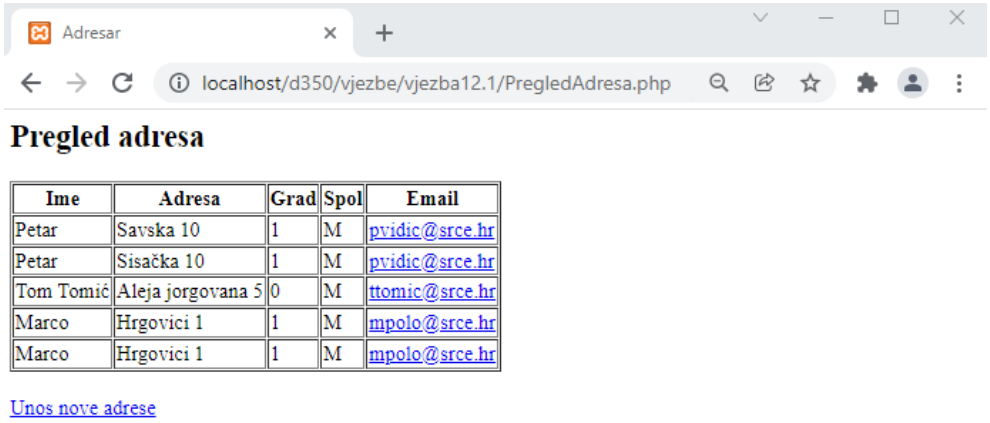
2. U datoteku, unutar PHP oznaka, unijet ćemo sljedeću sintaksu:

```
$email= $_POST['email'];
$naslov = $_POST['naslov'];
$tekst = $_POST['tekst'];

echo "Primatelj: $email<br/>";
echo "Naslov: $naslov<br/>";
echo "Tekst poruke: $tekst<br/><br/>";

ini_set("SMTP", "baltazar.srce.hr");
ini_set("sendmail_from", "ime.prezime@srce.hr");

if (mail($email, $naslov, $tekst))
{
  echo "Poruka je uspješno poslana.<br/>";
}
else
{
   echo "Poruka nije uspješno poslana.<br/>";
}
```

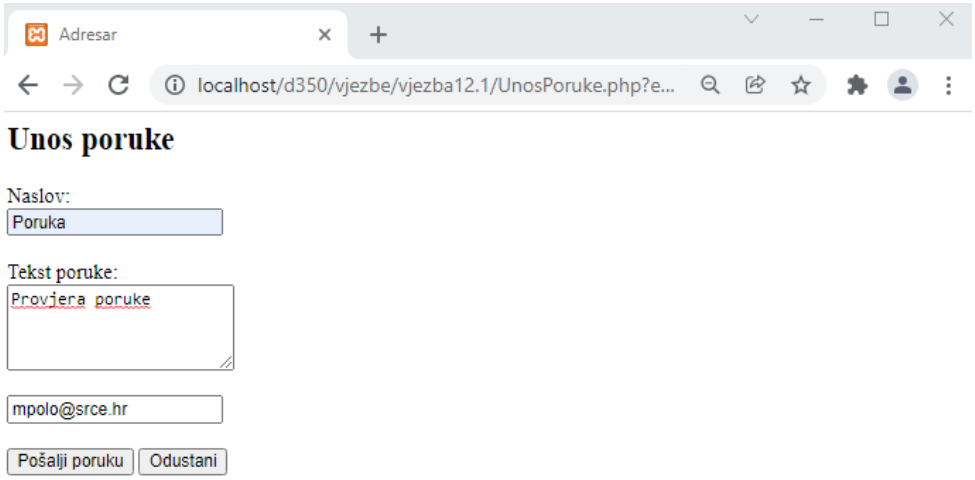
   Ako se ne koristi lokacija Srca za slanje poruka, potrebno je promijeniti poslužitelj *baltazar.srce.hr* na poslužitelj teleoperatera na lokaciji, npr. T-coma – *mail.t-com.hr.*

3. Spremimo datoteku.

4. Pokrenite program Brackets. U mapi *„…\D351\vjezbe\vjezba12.1*“ otvorite datoteku *PregledAdresa.php*.

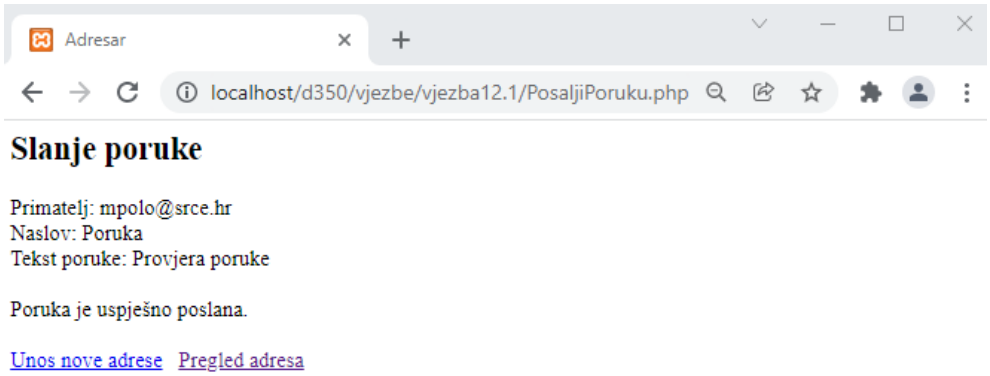


5. Pritiskom miša na *e-mail* adresu kontakta osobe otvorit će se obrazac za unos poruke. Popunimo obrazac proizvoljnim vrijednostima.



Primjećujemo da se u obrascu automatski popunilo polje *Primatelj* zato što je pritiskom na *e-mail* adresu u obrascu *PregledAdresa.php* postavljen prijenos *e-mail* adrese između obrazaca.

6. Pritisnemo *Pošalji poruku* i, ako je sve postavljeno u redu, dobijemo ispis da je poruka uspješno poslana.



13. Sesije i autentikacija korisnika

5. dan

Trajanje poglavlja:

60 min

Po završetku ovoga poglavlja moći ćete:

- definirati *kolačiće* i njihovo korištenje za spremanje malih količina podataka na klijentskom računalu
- objasniti sesiju između klijenta i poslužitelja
- opisati načine prenošenja identifikatora sesije – u kolačiću ili u URL-u
- opisati varijable sesije
- definirati započinjanje i završavanje sesije
- opisati autentikaciju korisnika pomoću korisničkog imena i lozinke.

Jedan od nedostataka HTTP protokola je što se između dvaju 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 stvara se sesija (*session*). Sesija započinje spajanjem korisnika (klijenta) na poslužitelj, a završava njegovim odlaskom (nakon što protekne određeno vrijeme, najčešće 24 minuta od njegove zadnje akcije).

13.1. Kolačići

Jedan od načina za ostvarivanje sesija 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 korisnikovu računalu)
4. pri sljedećem zahtjevu poslanom na isti poslužitelj klijent šalje kolačić povratno na poslužitelj

Kolačići se često koriste tako da se unutar kolačića zapisuje identifikator koji dobiva korisnik (odnosno identifikator sesije 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 za spremanje raznih korisničkih postavki (npr. odabir jezika stranica koje će se pretraživati u tražilici 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; opcijski)
- datum do kada *kolačić* vrijedi (Unixova vremenska oznaka
- opcijski) – nakon tog datuma kolačić će biti izbrisan s korisnikova računala
- putanja (znakovni niz; opcijski) – putanja (dio URL-a nakon domene) za koju vrijedi kolačić
- domena (znakovni niz; opcijski) – internetska domena poslužitelja; kolačić će biti poslan natrag ako domena i putanja odgovaraju URL-u na koji se šalje zahtjev
- sigurnost (cijeli broj; opcijski) – ako je ovaj argument 1, kolačić će se slati samo putem sigurne (HTTPS) veze

Napomena

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 kriptiranom obliku.

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 putem globalno dostupnog polja *$_COOKIE*.

Ovako bi poslužitelj pristupio kolačiću čiji je naziv „korisnik“:

```
<?php
  $kupac = $_COOKIE["korisnik"];
  echo $kupac;
?>
```

```
Ivica Ivić
```

Primjer uporabe kolačića primijenit ćemo na datoteci *zapisiKolacice.php*.

U datoteku ćemo upisati slijedeće 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.

Poslije oznaka za PHP kôd dodajte poveznicu na stranicu *procitajKolacice.php*:

```
?>
<a href="procitajKolacice.php">Pročitaj kolačiće</a>
```

Spremite datoteku u mapu „…\D351\primjeri\poglavlje13“ .

Za provjeru kolačića koristit ćemo datoteku *procitajKolacice.php*.

U datoteku ćemo dodati sljedeće naredbe: :

```
<?php
  foreach ($_COOKIE as $cookie)
  {
     echo $cookie . "<br/>";
  }
?>
```

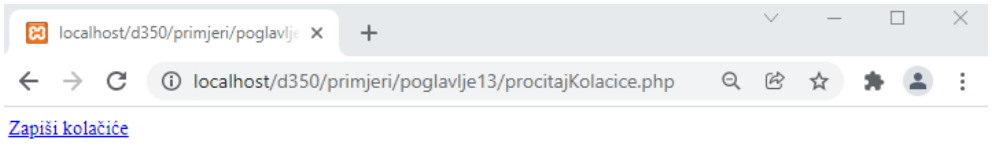
Skripta *procitajKolacice.php* služit će za čitanje i ispis vrijednosti *kolačića*. U polju *$_COOKIE* nalazit će se svi kolačići koje je klijent poslao poslužitelju.

Nakon oznaka za PHP kôd dodajte poveznicu na stranicu *zapisiKolacice.php*:

```
?>
<a href="zapisiKolacice.php">Zapiši kolačiće</a>
```

Spremite datoteku u mapu „…\D351\primjeri\poglavlje13“.

U *web*-pregledniku otvorit ćemo datoteku:

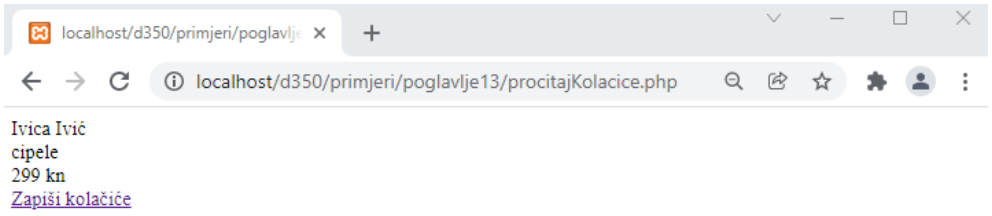


http://localhost/primjeri/poglavlje13/procitajKolacice.php .

Kolačići još nisu poslani klijentu, pa skripta *procitajKolacice.php* neće pronaći u polju *$_COOKIE* nijedan *kolačić*.

Odabirom poveznice *Zapiši kolačiće* izvršit će se skripta *zapisiKolacice.php* i klijentu će se poslati tri *kolačića* koja će on zapisati lokalno.

Nakon što odaberemo poveznicu *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*.

13.2. Sesije

Poslužitelj ostvaruje sesiju (spoj, sjednicu) s pojedinim klijentom kao niz HTTP zahtjeva i odgovora među njima. Svaka sesija dobiva svoj jedinstveni identifikator, na temelju čega se prepoznaje da pojedini zahtjevi pripadaju točno određenoj sesiji.

Identifikator sesije stvara se na poslužitelju i šalje klijentu unutar HTTP odgovora. Klijent ga potom prosljeđuje natrag prilikom svakog sljedećeg zahtjeva.

Sesija će trajati sve dok stižu zahtjevi klijenta. Ako od posljednjeg zahtjeva protekne određeno vrijeme (u PHP-u je predefinirana vrijednost tog vremena 24 minute), sesija će se automatski zatvoriti.

Osnovni način za prosljeđivanje identifikatora sesije i njegovo čuvanje na klijentskom računalu su kolačići. Ako su kolačići onemogućeni ili nisu podržani u korisnikovu *web*-pregledniku, identifikator sesije će se prosljeđivati putem URL-a. Primjer takvog URL-a:

http://www.srce.hr?PHPSESSID=242c489fb4a1bc79f5cf365988167e4d

Sesije se mogu upotrijebiti za spremanje podataka koji su vezani za sesiju, dakle za pojedinog korisnika. Takvi podaci spremaju se u varijable sesije koje će biti vidljive samo za tog korisnika i trajat će samo dok traje i sesija.

Varijable sesije

Napomena

Vrijednosti varijabli sesije spremaju se, ako nije drugačije postavljeno, u datotekama u mapi */tmp* na poslužitelju.

Varijablama sesija pristupa se putem globalno dostupnog polja *$_SESSION*. Stvaranje nove varijable sesije zapravo je dodavanje novog člana u polje *$_SESSION*:

```
<?php
  $_SESSION["korisnik"] = $imeKorisnika;
?>
```

Čitanje vrijednosti varijable sesije je pristupanje članu polja *$_SESSION* pomoću odgovarajućeg znakovnog ključa:

```
<?php
  if (isset($_SESSION["korisnik"]))
  {
    echo $_SESSION["korisnik"];
   }
?>
```

Napomena

Ako se na poslužitelju vrijednost *session.auto_start* postavi na 1, sesija će automatski započinjati po primanju prvog klijentova zahtjeva. Predefinirana vrijednost ove postavke je 0.

Za provjeru je li neka varijabla sesije stvorena može se koristiti funkcija *isset*.

Varijable sesije čuvaju se na poslužitelju. Putem identifikatora sesije moguće je pristupiti varijablama koje pripadaju pojedinoj sesiji. Varijable sesije za svakog će korisnika sadržavati različite vrijednosti.

Uz podatke poslane unutar URL-a (metodom GET) ili putem obrasca (metodom *POST*), korištenje varijabli sesije još je jedan način čuvanja podataka između dvaju HTTP zahtjeva.

Otvaranje sesije

Napomena

Ako se više puta postavi funkcija za otvaranje sesije *session_start*, PHP može javiti grešku.

Zato je potrebno obratiti pažnju da se funkcija pokretanja sesije obavi samo jednom unutar skripte.

Za započinjanje sesije, potrebno je pozvati funkciju ***session_start***. Funkcija prima opcijskiargument (asocijativni niz) za promjenu trenutnih postavki sesija i uvijek vraća vrijednost TRUE.

Čak i ako je sesija već započeta, funkciju *session_start* potrebno je pozvati na početku svake skripte koja koristi varijable sesije jer se, kad se ona pozove, varijable sesije učitavaju u polje *$_SESSION*.

Primjer poziva funkcije:

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

Zatvaranje sesije

Sesija će se automatski zatvoriti nakon što protekne zadani vremenski rok od posljednjeg zahtjeva klijenta. No, sesiju 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 sesije bit će izgubljene.

13.3. Autentikacija korisnika

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 njega se traži unos korisničkog imena i lozinke. Nakon toga provjerava se 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 često se ograničava.)

Ako je korisnik upisao odgovarajuće ime i lozinku, omogućit će mu se pristup ostatku aplikacije. Na ovom mjestu će se najčešće, korištenjem varijable sesije, zapisati da je autentikacija korisnika izvršena. Zahvaljujući tome, korisnik neće morati ponovno upisivati lozinku prilikom svakog novog zahtjeva.

Kad sesija istekne, korisnik će se morati ponovno autenticirati upisivanjem korisničkog imena i lozinke.

Postavlja se pitanje je li moguće da korisnik podastre lažni identifikator sesije (koji može promijeniti izmjenom kolačića ili URL-a) i na taj se način predstavi kao već autenticirani korisnik. Ta je mogućnost matematički vrlo malo vjerojatna zato što svaka PHP sesija dobiva nasumični identifikator, koji se zatim kriptira.

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 sesije), 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.

Primjer autentikacije korisnika napravit ćemo otvaranjem skripte *prijava.php* iz mape *"…\D351\primjeri\poglavlje13*".

U datoteku ćemo prije oznaka za HTML dodati slijedeći kôd:

Napomena

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.

Napomena

Jedna od mogućnosti za neovlašteni pristup aplikaciji je tzv. *session hijacking*. Ako napadač presretne klijentov HTTP zahtjev, iz njega može preuzeti identifikator sesije 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 danas umjesto HTTP protokola koristi HTTPS protokol.

```
<?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!";
      echo $poruka;
    }
  }
?>
```

Prva naredba (poziv funkcije *session_start*) započinje sesiju (ako je nije već započeta) i učitava vrijednosti varijabli sesije u polje *$_SESSION*.

Zatim se funkcijom *isset* provjerava jesu li metodom POST poslani korisničko ime i lozinka. Ako jesu, provjerit će 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 zapisat će se u varijablu *sesije*. Naziv te varijable *sesije*, odnosno njen znakovni ključ u polju *$_SESSION*, bit će "korisnickoIme". Potom će, korištenjem funkcije *header* korisnik biti preusmjeren na stranicu *index.php*.

Datoteku *indeks.php* koristit ćemo kao početnu stranicu u kojoj će funkcija *isset* provjeravati je li sesija otvorena i, ako nije, preusmjerit će XXXX na datoteku *prijava.php*.

```
<?php
  session_start();

  $prijava="";

  if (!isset($_SESSION["korisnickoIme"]))
  {
    header("Location: prijava.php");
  }

?>
```

Još je potrebno dodati i datoteku *odjava.php* koja će koristiti korisnicima ako se žele samostalno odjaviti sa stranice:

```
<?php

  session_start();
  session_destroy();
```

```
   header("Location: index.php");

?>
```

Da bi se sesiju moglo koristiti u skripti, potrebno je najprije pozvati funkciju *session_start*. Nakon toga poziva se funkcija *session_destroy*, koja će zatvoriti sesiju i izbrisati sve varijable sesije. Zatim će se korisnik preusmjeriti na stranicu *index.php*.

Vježba 13.1 – Autentikacija korisnika iz tablice u bazi

1. Pokrenite program Brackets i otvorite datoteku *prijava.php* u mapi "…\D351\vjezbe\vjezba13.1" .

2. U datoteci se nalazi obrazac koji 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. U skriptu ćemo postaviti PHP kôd za otvaranje sesije i uključiti datoteku za povezivanje s bazom podataka:

```
<?php

   session_start();
   include "otvoriVezu.php";
?>
```

4. Nakon toga dodat ćemo PHP kôd za provjeru jesu li u obrascu uneseni korisničko ime i lozinka:

```
<?php

   include "otvoriVezu.php";
   if (isset($_POST["korisnickoime"]) &&
   isset($_POST["lozinka"]))
         {

         }
?>
```

5. Sada slijedi pridruživanje varijabli elementima poslanim s obrasca:

```
<?php

   if (isset($_POST["korisnickoime"]) &&
   isset($_POST["lozinka"]))
         {
            $korisnickoIme = $_POST["korisnickoime"];
            $lozinka = $_POST["lozinka"];
            $korisnik='';
            $pass='';
         }
?>
```

Postavljanje varijabli *$korisnik i $pass* na početnu praznu vrijednost nije nužno, ali je poželjno kako ne bi došlo do pogrešaka kod pokretanja skripte u slučaju da u obrazac nisu uneseni podaci.

6. Kako bismo usporedili jesu li podaci uneseni u obrazac jednaki podacima korisnika koji se nalaze u tablici u bazi, moramo dohvatiti podatke iz tablice:

```
<?php

          $korisnickoIme = $_POST["korisnickoime"];
          $lozinka = $_POST["lozinka"];
          $korisnik='';
          $pass='';

        $rezultat = mysqli_query($veza,"SELECT * FROM korisnici
                    where username='$korisnickoIme'");
        while ($redak = mysqli_fetch_array($rezultat))
           {
              $korisnik= $redak['username'];
              $pass= $redak['password'];
           }
       }
?>
```

7. Sada slijedi usporedba podataka unesenih u obrazac s podacima iz tablice:

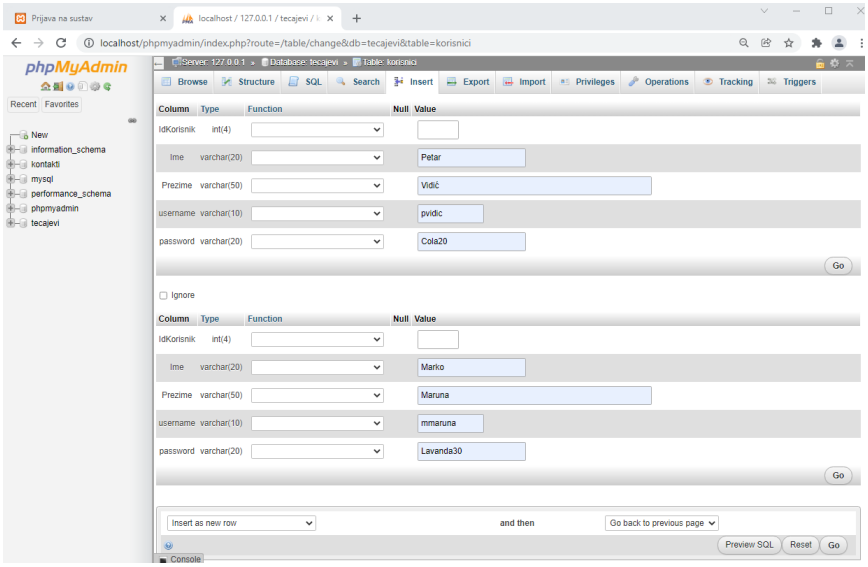
```
<?php
          {
              $korisnik= $redak['username'];
              $pass= $redak['password'];
          }
          if($korisnickoIme ==$korisnik && $lozinka ==$pass)
             {
                $_SESSION["korisnickoime"] = $korisnickoIme;
                header("Location: izbornik.php");
             }
             else
             {
                $poruka = "Neuspješna prijava";
                echo $poruka;
             }

             mysqli_free_result($rezultat);

       }
?>
```

8. Spremite izmjene datoteke u mapu „*…\D351\vjezbe\vjezba13.1*“.

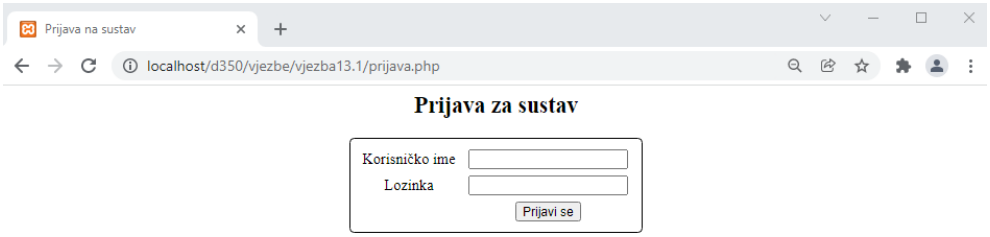
9. Otvorimo aplikaciju *phpMyAdmin* s lokacije http://localhost/phpMyAdmin/.

10. Ako već nije stvorena, stvorite novu tablicu *korisnici* u bazi *tecajevi* s poljima: *Ime, Prezime, Username, Password.* Samostalno odredite tipove podataka za ta polja. Korisničko ime mora biti jedinstveno.

11. Otvorimo bazu *tecajevi* i u tablicu *korisnici* putem naredbe *Insert* unesemo korisnika po želji i odaberemo *Go.*

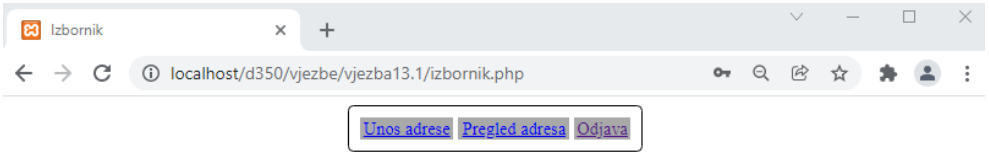


12. Nakon što ste pohranili podatke o korisnicima, u *web*-preglednik upišite adresu: http://localhost/D351/vjezbe/vjezba13.1/prijava.php

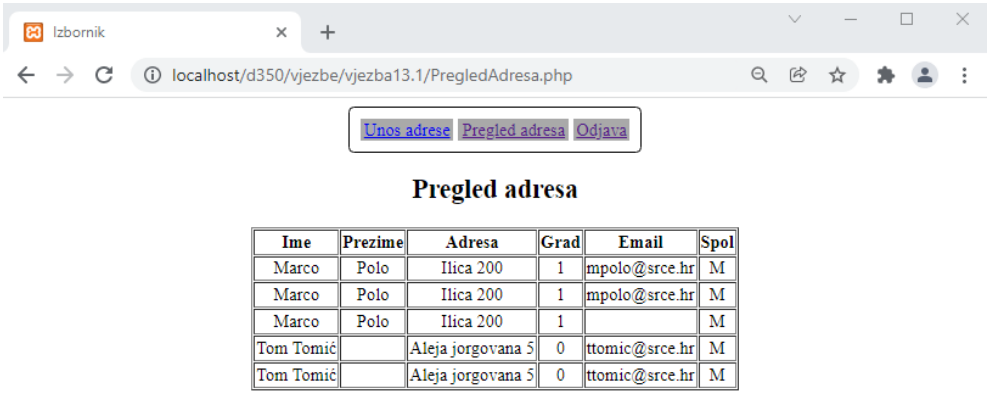


13. Unesimo podatke za prijavu (bilo koji od podataka koji smo prethodno pohranili u tablicu *korisnici* u bazi *tecajevi* i odaberemo *Prijavi se*.

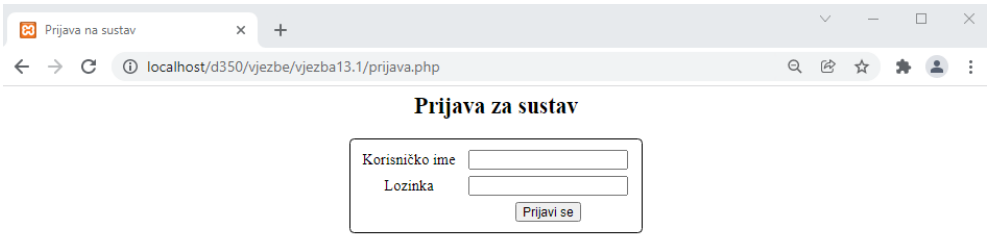
14. Ako su podaci za prijavu točni, otvorit će se stranica *izbornik.php*.



15. Sada se možemo nesmetano kretati po stranici pomoću izbornika.



16. Kada želimo završiti, dovoljno je pritisnuti na *Odjava* i sustav će nas vratiti na *prijava.php*.



14. Uvod u objektno orijentirani model

5. dan

Trajanje poglavlja:

120 min

Po završetku ovoga poglavlja moći ćete:

- opisati osnove objektno orijentiranog programiranja
- razlikovati klase i objekte
- definirati klase i objekte u PHP skripti
- opisati stvaranje konekcije prema bazi u objektno orijentiranom modelu
- objasniti dohvat podataka iz tablice po objektno orijentiranom modelu.

U prijašnjim poglavljima koristili smo proceduralno programiranje, a u ovom poglavlju napravit ćemo kratak uvod u objektno orijentirano programiranje.

Proceduralno programiranje jest model izveden iz strukturnog programiranja prema kojem se često ponavljan blok naredbi izdvaja u procedure čije se izvršavanje izvodi pozivom procedure u bilo kojem trenutku.

Objektno orijentirano programiranje je model koji koristi klase i objekte koji sadržavaju podatke u obliku atributa i kodove u obliku metoda. Dostupno je u PHP-u tek od verzije 5.0.

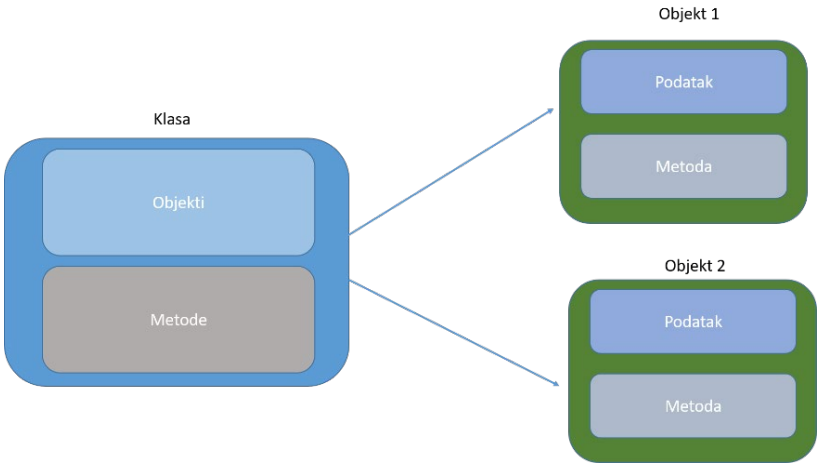
Objektno orijentirano programiranje ima određenih prednosti u odnosu na proceduralno:

- brže je i jednostavnije za izvršavanje
- omogućava jasnu strukturu programa
- kod je jednostavniji za održavanje, modificiranje i otkrivanje grešaka

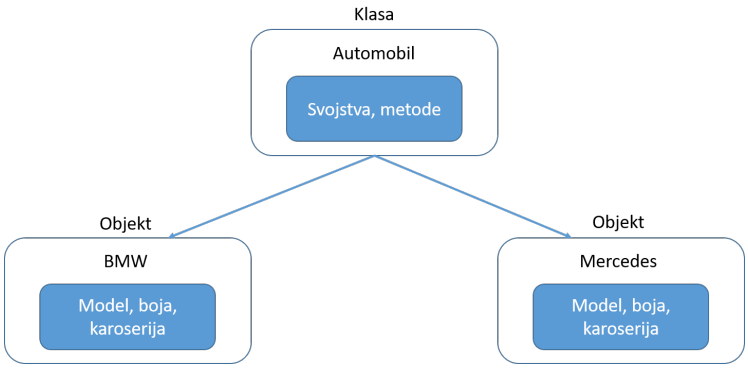
14.1. Objekti i klase

Već smo spomenuli da objektno orijentirano programiranje koristi klase i objekte. Računalni programi dizajnirani su da koriste koncept objekata koji komuniciraju sa stvarnim svijetom.

Primjeri objekata i klasa:



Prikaz objekata i klasa na primjeru automobila:



Dakle, klase i objekti su dva glavna aspekta objektno orijentiranog programiranja. Klasa je predložak objekata, a objekti su instancije klase.

14.1.1. Klase

U stvarnom svijetu možemo pronaći razne oblike pojedinih objekata, Na primjer, postoje osobni automobili raznih tipova, vrsta i izgleda. Zato svaki objekt možemo opisati sa njegovim svojstvima, a sve slične objekte postaviti u zajedničku klasu.

Dakle, iz jedne klase možemo dobiti mnogo sličnih objekata koji će svaki imati svoja zasebna svojstva, a to bi bio i redoslijed na kojim se definiraju objekti i klase u objektno orijentiranom programiranju.

14.1.2 Objekti

U svijetu oko nas svakodnevno smo okruženi raznim materijalnim objektima poput računala, mobilnih telefona, automobila. Osim toga, možemo primijetiti i nematerijalne objekte poput bankovnih računa i transakcija.

Svi ti objekti imaju dvije zajedničke karakteristike: stanje i ponašanje. Objekt pohranjuje svoje stanje u varijable koje se često nazivaju ili opisuju kao svojstva. Objekt također može prikazati svoje ponašanje putem funkcija poznatijih kao metode.

14.2. Definiranje objekata i klasa u skriptama

U ovom primjeru prikazat ćemo kako u PHP-u definirati klase i objekte.

Stvorit ćemo novu PHP datoteku pod nazivom *primjer1.php* (mapa …D351\primjeri\poglavlje14\)

Klase se definiraju uporabom funkcije ***class*** i dodjelom imena:

```
<?php
   class Automobil
         {

         }
?>
```

Unutar vitičastih zagrada definiraju se svojstva i metode klase. Funkcija ***public*** određuje vidljivost svojstva, tj. u ovom slučaju deklarira varijablu svojstva objekta:

```
<?php
   class Automobil
         {
         public $naziv;
         public $color;
?>
```

Nakon toga ćemo definirati i metode koje će se koristiti. U ovom primjeru definiramo metodu ***set_name*** kojom ćemo postaviti naziv objekta i metodu ***get_name*** koja će napraviti dohvat postavljenog naziva objekta:

```
<?php
         public $color;

         function set_name($naziv)
            {
              $this->naziv = $naziv;
            }
         function get_name()
            {
              return $this->naziv;
            }
         }
?>
```

U prethodnom primjeru navedene su uobičajene konvencije koje je moguće vidjeti u mnogim programskim jezicima kao što su Java i Ruby. Uporabom funkcija *Set* i *Get* definiraju se metode za postavljanje“ i dohvat svojstava na klasama ili objektima.

Ugrađena varijabla ***$this*** (ugrađena u svim objektima) ukazuje na trenutni objekt. ***$this*** je posebna varijabla koja koristeći samoreferenciju ukazuje na pristup svojstvima i pozivanje metoda trenutne klase.

Nakon što smo definirati svojstva i metode koje će se koristiti, definiramo i objekte nad kojima će se te metode provoditi:

```
<?php
    $Audi = new Automobil();
    $Toyota = new Automobil();
    $Audi->set_name('Audi');
    $Toyota->set_name('Toyota');

?>
```

Popis objekata dobit ćemo ispisom naziva objekata, tj. varijabli kojima su pridruženi nazivi:

```
<?php
    echo $Audi->get_name();
    echo "<br>";
    echo $Toyota->get_name();
?>
```

Ako želimo dodati još neka svojstva objektu, poput npr. boje, dodat ćemo funkcije s postavljanjem boja i dohvatom boja:

```
            return $this->naziv;
          }

    function set_color($color)
          {
            $this->color = $color;
           }
    function get_color()
          {
           return $this->color;
          }
     }
     $Audi = new Automobil();
     $Toyota = new Automobil();
```

Definiranje boja napravimo pridruživanjem vrijednosti varijablama:

```
    $Audi->set_name('Audi');
    $Toyota->set_name('Toyota');

    $Audi->set_color('Red');
    $Toyota->set_color('Blue');

    echo $Audi->get_name();
    echo "<br>";
```

Na kraju još dodamo ispis boja :

```
    echo $Audi->get_name();
    echo "<br>";
    echo $Audi->get_color();
    echo "<br>";
    echo $Toyota->get_name();
    echo "<br>";
    echo $Toyota->get_color();
    echo "<br>";
?>
```

Dosad smo u primjerima prvo definirali objekte te im nakon toga dodjeljivali određena svojstva. Sada ćemo u skriptu dodati funkciju ***_construct*** . Funkcija ***_construct*** omogućava da se svojstva objekta inicijaliziraju nakon što se kreira objekt. Funkcija ***_construct*** nas na taj način oslobađa pozivanja funkcije ***set_name*** i istodobno smanjuje količinu koda u skripti. PHP će automatski pozvati ovu funkciju kada se kreira objekt.

U sljedećem primjeru napravit ćemo skriptu koja ispisuje nazive i boje automobila na način da uključimo funkciju ***_construct****:* (datoteka *primjer2.php,* mapa …D351\primjeri\poglavlje14\)

```
<?php
class Automobil
        {
        public $naziv;
        public $boja;

        function __construct($naziv, $boja)
           {
```

```
            $this->naziv = $naziv;
            $this->boja = $boja;
          }/*
        function set_name($naziv)
          {
            $this->naziv = $naziv;
          }*/
        function get_name()
          {
            return $this->naziv;
          }/*
        function set_color($boja)
          {
            $this->color = $boja;
           }*/
         function get_color()
          {
           return $this->boja;
          }
        }
      $Audi = new Automobil("Audi", "red");
      echo $Audi->get_name();
      echo "<br>";
      echo $Audi->get_color();
      echo "<br>";
      $BMW = new Automobil("BMW", "blue");
      echo $BMW->get_name();
      echo "<br>";
      echo $BMW->get_color();
?>
```

14.3. Povezivanje s bazom podataka

Povezivanje s bazom podataka u objektno orijentiranom programiranju slično je proceduralnom. Pojednostavljene su pojedine funkcije koje olakšavaju povezivanje.

Stvorit ćemo novu PHP datoteku pod nazivom *otvoriVezu.php* (mapa …D351\primjeri\poglavlje14\) i unutar PHP oznaka dodati kôd za povezivanje na poslužitelj i bazu *tecajevi* .

```
<?
  session_start();
  $servername="localhost";
  $username="root";
  $password="";
  $database="tecajevi";

  $veza = new
          mysqli($servername,$username,$password,$database);

      if (mysqli_connect_errno())
       {
        echo "Neuspjelo spajanje na poslužitelj.";
        echo mysqli_connect_error();
        exit;
       }
       else
```

```
        {
         echo "Spojeni ste na poslužitelj";
        }
?>
```

Varijable na početku skripte definiraju parametre povezivanja na poslužitelj i bazu. Varijabla ***$veza*** definira novi ***mysqli*** objekt i pokreće povezivanje na poslužitelj koristeći parametre varijabli navedenih na početku. Da bismo bili sigurni da je povezivanje uspjelo, dodali smo ***if,*** uvjetnu provjeru povezivanja koja u slučaju neuspjeha prekida skriptu i javlja pogrešku spajanja, a u slučaju da je spajanje uspješno poslat će potvrdnu poruku.

14.4. Dohvat podataka iz tablice

Kada smo otvorili vezu prema poslužitelju, možemo napraviti dohvat, unos i ažuriranje podataka iz tablice koristeći PHP-ov objektno orijentirani model

Otvorit ćemo PHP datoteku pod nazivom ***PregledAdresa.php*** (mapa …D351\primjeri\poglavlje14\)

Unutar PHP oznaka uključit ćemo skriptu koja ***otvoriVezu.php*** i dodat ćemo kôd za ispis podataka iz tablice *polaznik* .

```
<?
    include "otvoriVezu.php";
    echo "<br/>";
    $query = "SELECT * FROM polaznik limit 20";
    if ($result= $veza->query($query))
    {
        while ($row=$result->fetch_assoc())
        {
            echo $row['Ime'].' '.$row['Prezime'];
            echo "<br/>";
        }
    }
    $veza->close();
?>
```

U prvom retku otvara se veza prema bazi podataka. U trećem redu definiramo varijablu ***$query*** kojoj ćemo pridružiti upit za dohvat podataka iz tablice *polaznik* limitiranih na prvih 20. U provjeru rezultata uključili smo petlju *while* koja će ispisati imena i prezimena prvih 20 polja tablice *polaznik*. Na kraju ćemo dodati zatvaranje veze.

Vježba 14.1 – PHP OO model otvaranje veze

1. Pokrenite program Brackets i stvorite datoteku *otvoriVezu.php* u mapi "…\D351\vjezbe\vjezba14.1" .
2. U skriptu ćemo postaviti PHP kôd za otvaranje sesije i stvoriti datoteku za povezivanje s bazom podataka.
3. Definirat ćemo varijable potrebne za povezivanje:

```
<?php
  session_start();
  $posluzitelj = "localhost";
  $korisnik = "root";
  $lozinka = "";
  $baza = "tecajevi";
?>
```

4. Nakon toga definirat ćemo novu *mysqli* varijablu koja će pokrenuti otvaranje veze s parametrima prethodnih varijabli:

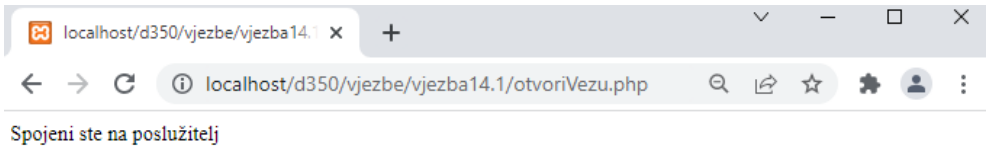
```
<?php

  $lozinka = "";
  $baza = "tecajevi";

  $veza = new mysqli($posluzitelj,$korisnik,
  $lozinka,$baza);

?>
```

5. Nakon toga definirat ćemo uvjetnu provjeru postavljanja veze i unutar nje definirati poruke u slučaju pogreške ili uspješnog spajanja:

```
<?php

  $veza = new mysqli($posluzitelj,$korisnik,
  $lozinka,$baza);

  if (mysqli_connect_errno())
    {
     echo "Neuspjelo spajanje na poslužitelj.";
     echo mysqli_connect_error();
     exit;
    }
    else
    {
     echo "Spojeni ste na poslužitelj";
    }
?>
```

6. Spremite datoteku u mapu „*…\D351\vjezbe\vjezba14.1*“.

7. Nakon što smo pohranili podatke o korisnicima, u *web*-preglednik upišite adresu: http://localhost/vjezbe/vjezba14.1/otvoriVezu.php



Vježba 14.2 – Dohvat podataka iz tablice

1. U ovoj vježbi potrebno je ispisati sve predavače i odjele kojima pripadaju.

2. Pokrenite program Brackets i otvorite datoteku *PregledAdresa.php* u mapi „…\D351\vjezbe\vjezba14.2“ .

3. U skriptu ćemo unutar PHP kôda dodati varijable kojima ćemo pridružiti SQL upit za dohvat podataka iz tablica. Da bismo u ovom slučaju prikazali kojim odjelima pojedini predavači pripadaju, moramo povezati tablice *predavac* i *odjel* i prikazati polja *Ime, Prezime i odjel* predavača.

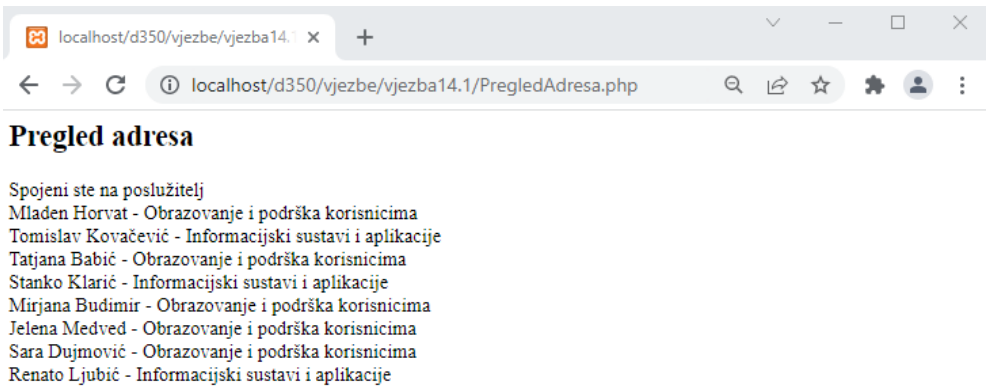
```
<?php
  include "otvoriVezu.php";
  echo "<br/>";
  $query = "SELECT Ime, Prezime, Naziv FROM predavac inner join
odjel on predavac.odjelid = odjel.id";
?>
```

4. Dodat ćemo provjeru ispisa i unutar provjere petlju s definiranim poljima *Ime, Prezime i Naziv* (Odjela) koja želimo ispisati i na kraju zatvoriti vezu:

```
   if ($result= $veza->query($query))
        {
            while ($row=$result->fetch_assoc())
            {
                echo $row['Ime'].' '.$row['Prezime'].' -
  '.$row['Naziv'];
                echo "<br/>";
            }
        }
        $veza->close();
   ?>
```

5. Spremite datoteku u mapi „*…\D351\vjezbe\vjezba14.2*“.

6. Nakon što ste pohranili podatke o korisnicima, u *web*-preglednik upišite adresu: http://localhost/vjezbe/vjezba14.2/PregledAdresa.php/

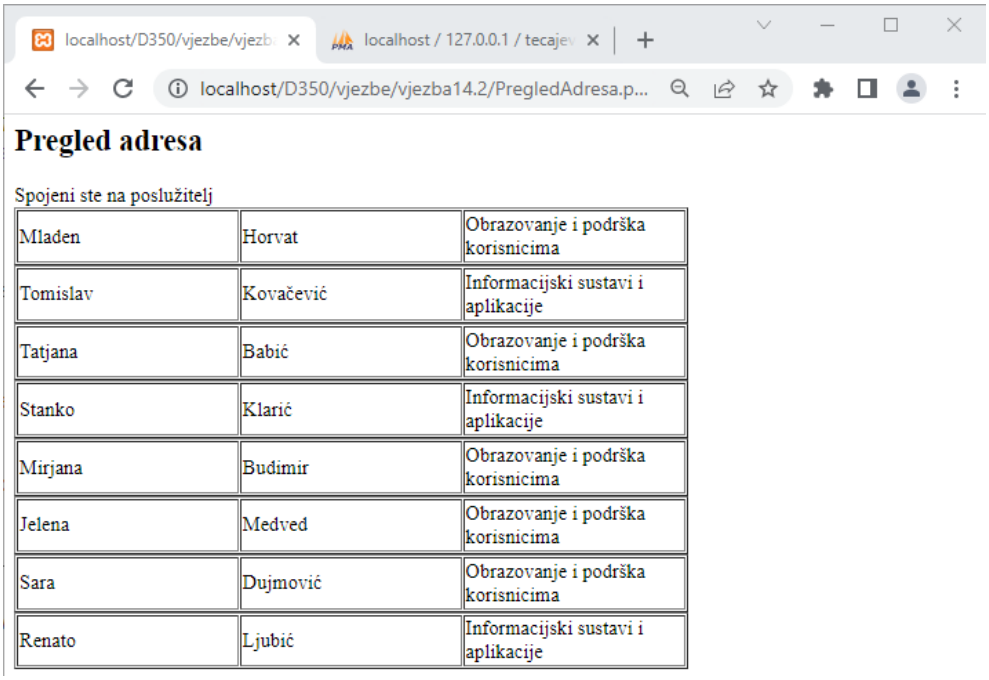


7. Uredit ćemo prikaz ispisa tako da se *Ime, Prezime* i *Naziv* prikažu unutar HTML tablice:

```
   if ($result= $veza->query($query))
        {
```

```
            while ($row=$result->fetch_assoc())
            {
                echo "<table border='1'>";
                echo "<tr>";
                echo "<td>".$row['Ime']."</td>";
                echo "<td>".$row['Prezime']."</td>";
                echo "<td>".$row['Naziv']."</td>";
                echo "</tr>";
                echo "</table>";          }
        }
        $veza->close();
    ?>
```

8. Provjerimo izgled skripte u *web*-pregledniku otvaranjem adrese: *http://localhost/vjezbe/vjezba14.2/PregledAdresa.php:*



Pregled adresa

Spojeni ste na poslužitelj

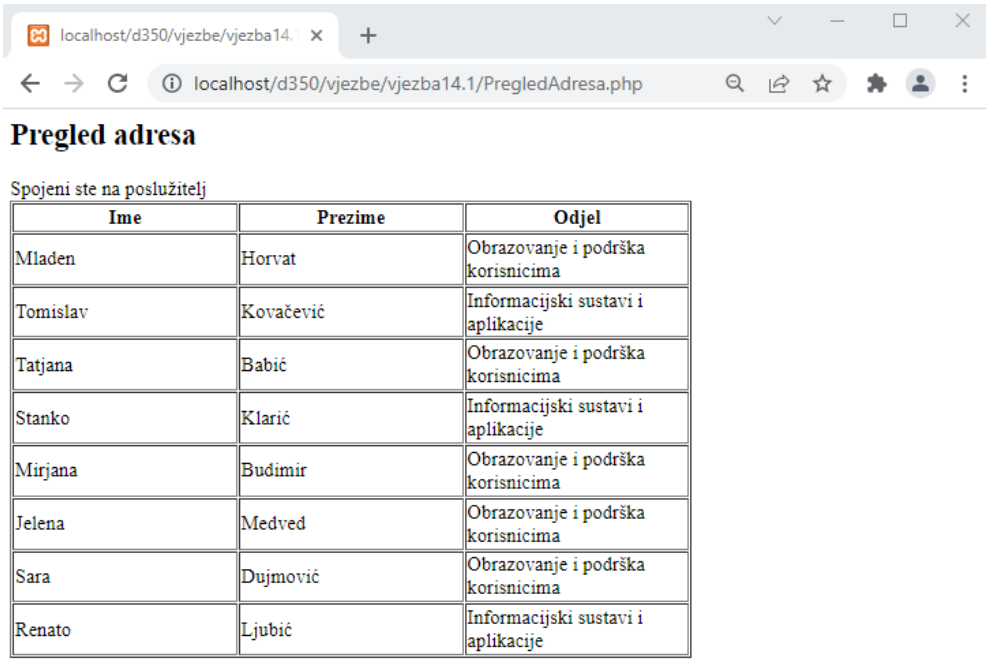
Mladen	Horvat	Obrazovanje i podrška korisnicima
Tomislav	Kovačević	Informacijski sustavi i aplikacije
Tatjana	Babić	Obrazovanje i podrška korisnicima
Stanko	Klarić	Informacijski sustavi i aplikacije
Mirjana	Budimir	Obrazovanje i podrška korisnicima
Jelena	Medved	Obrazovanje i podrška korisnicima
Sara	Dujmović	Obrazovanje i podrška korisnicima
Renato	Ljubić	Informacijski sustavi i aplikacije

9. Da bismo dobili izgled tablice sa zaglavljima, tj. naslovima stupaca, izmijenit ćemo skriptu:

```
    <h2>Pregled adresa</h2>
        <table border='1'>
            <tr>
                <th>Ime</th>
                <th>Prezime</th>
                <th>Odjel</th>
            </tr>
    <?php
       include "otvoriVezu.php";
        echo "<br/>";
        $query = "SELECT Ime, Prezime, Naziv FROM predavac inner
    join odjel on predavac.odjelid = odjel.id";
        if ($result= $veza->query($query))
        {
            while ($row=$result->fetch_assoc())
            {
                echo "<table border='1'>";
                echo "<tr>";
```

```
                echo "<td>".$row['Ime']."</td>";
                echo "<td>".$row['Prezime']."</td>";
                echo "<td>".$row['Naziv']."</td>";
                echo "</tr>";
                echo "</table>";
            }
        }
        $veza->close();


                echo "</table>";
    ?>
```

10. Provjerimo sada još izgled skripte u *web*-pregledniku otvaranjem adrese: http://localhost/vjezbe/vjezba14.2/PregledAdresa.php.



Ime	Prezime	Odjel
Mladen	Horvat	Obrazovanje i podrška korisnicima
Tomislav	Kovačević	Informacijski sustavi i aplikacije
Tatjana	Babić	Obrazovanje i podrška korisnicima
Stanko	Klarić	Informacijski sustavi i aplikacije
Mirjana	Budimir	Obrazovanje i podrška korisnicima
Jelena	Medved	Obrazovanje i podrška korisnicima
Sara	Dujmović	Obrazovanje i podrška korisnicima
Renato	Ljubić	Informacijski sustavi i aplikacije

Vježba 14.3 – Dohvat podataka iz tablice po uvjetu

1. U ovoj vježbi potrebno je ispisati sve predavače i odjele kojima pripadaju.

2. Pokrenite program Brackets i kopirajte datoteku *PregledAdresa.php* u mapi „…\D351\vjezbe\vjezba14.2“ u mapu „…\D351\vjezbe\vjezba14.3“ .

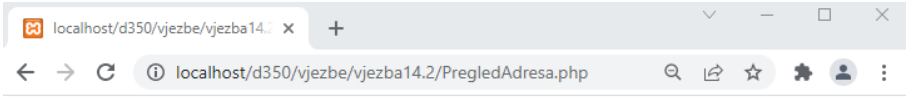
3. U skripti ćemo izmijeniti PHP kôd kako da bismo prikazali prvih 20 polaznika i gradove kojima pripadaju. U ovom slučaju moramo povezati tablice *polaznik* i *odjel* i prikazati polja *Ime, Prezime i Grad* u kojem živi polaznik:

```
<?php
  include "otvoriVezu.php";
  echo "<br/>";
  $query = "SELECT Ime, Prezime, Naziv FROM polaznik inner join
Grad on polaznik.gradid = grad.id limit 20";
?>
```

4. Za potrebe prikaza samo ćemo malo izmijeniti izgled postojeće skripte *PregledAdresa.php* ::

```
<h2>Pregled adresa</h2>
    <table border='1'>
        <tr>
            <th>Ime</th>
            <th>Prezime</th>
            <th>Grad</th>
        </tr>
<?php
   include "otvoriVezu.php";
    echo "<br/>";
    $query = "SELECT Ime, Prezime, Naziv FROM polaznik inner
join grad on polaznik.Gradid = grad.Id limit 20";
    if ($result= $veza->query($query))
    {
        while ($row=$result->fetch_assoc())
        {
            echo "<tr>";
            echo "<td>".$row['Ime']."</td>";
            echo "<td>".$row['Prezime']."</td>";
            echo "<td>".$row['Naziv']."</td>";
            echo "</tr>";
        }
    }
    $veza->close();


            echo "</table>";
?>
```

5. Provjerimo izgled ispisa u *web*-pregledniku otvaranjem adrese: http://localhost/vjezbe/vjezba14.3/PregledAdresa.php.



Pregled adresa

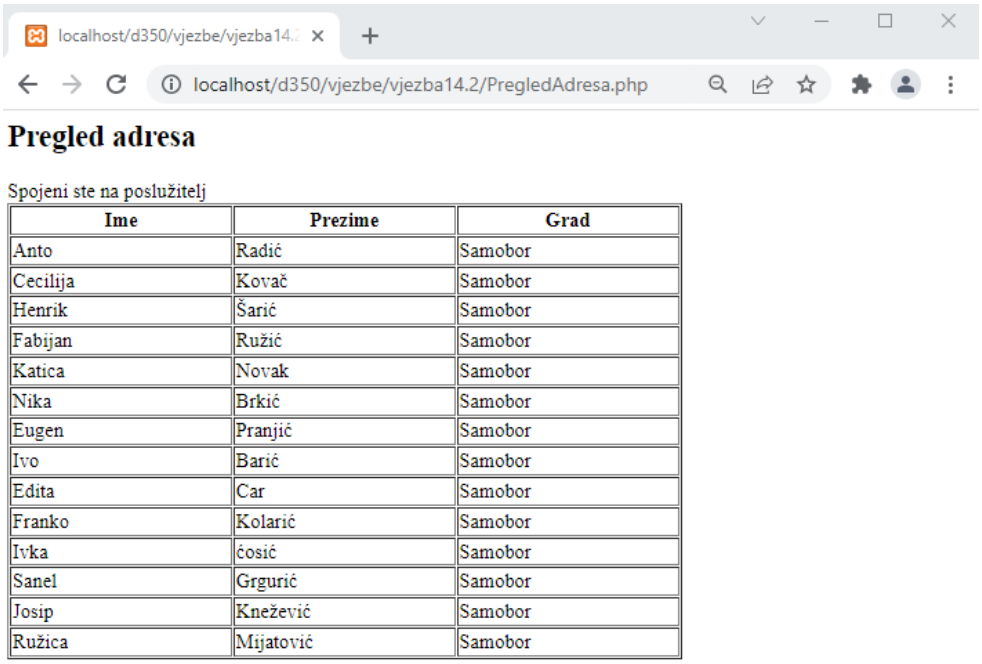
Spojeni ste na poslužitelj

Ime	Prezime	Grad
Igor	Vlahović	Zaprešić
Pavao	Šimunović	Zagreb
Jerko	Pavić	Zagreb
Adrijan	Marušić	Zagreb
Ivka	Mihaljević	Zagreb
Alojz	Ljubić	Zagreb
Anto	Radić	Samobor
Dijana	Ivić	Zagreb
Kristijan	Petričević	Zagreb
Mirta	Marjanović	Zagreb
Dorotea	Jurković	Zagreb
Danilo	Živković	Zagreb
Edi	Jelić	Zagreb
Gabriela	Marinović	Zagreb
Jasminka	Kovačević	Zagreb
Anka	Šimunić	Zagreb
Bojan	Mandić	Zagreb
Davorin	Topić	Zagreb
Cecilija	Kovač	Samobor
Nedjeljka	Abramović	Zagreb

6. Dodatno ćemo još prikazati samo one polaznike koji su iz Samobora:

```
<?php
  include "otvoriVezu.php";
  echo "<br/>";
  $query = "SELECT Ime, Prezime, Naziv FROM polaznik inner join
Grad on polaznik.gradid = grad.id where naziv = 'Samobor'";
?>
```

7. Provjerimo izgled ispisa u *web*-pregledniku otvaranjem adrese: http://localhost/vjezbe/vjezba14.3/PregledAdresa.php.

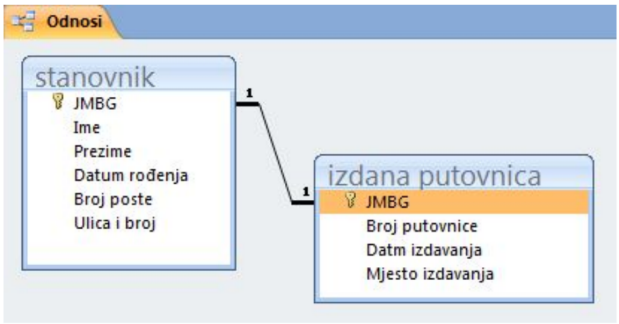


Ime	Prezime	Grad
Anto	Radić	Samobor
Cecilija	Kovač	Samobor
Henrik	Šarić	Samobor
Fabijan	Ružić	Samobor
Katica	Novak	Samobor
Nika	Brkić	Samobor
Eugen	Pranjić	Samobor
Ivo	Barić	Samobor
Edita	Car	Samobor
Franko	Kolarić	Samobor
Ivka	ćosić	Samobor
Sanel	Grgurić	Samobor
Josip	Knežević	Samobor
Ružica	Mijatović	Samobor

15. Dodatak

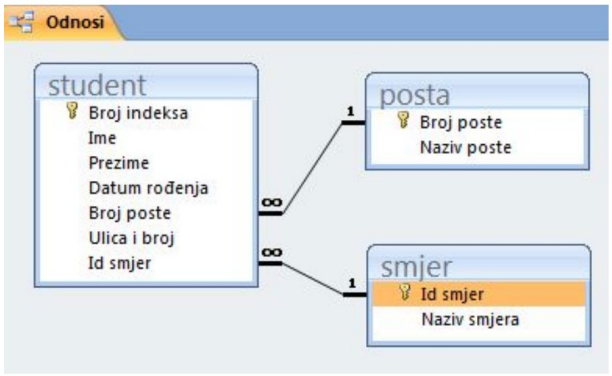
Odnosi u relacijskim bazama podataka:

Jedan prema jedan 1:1

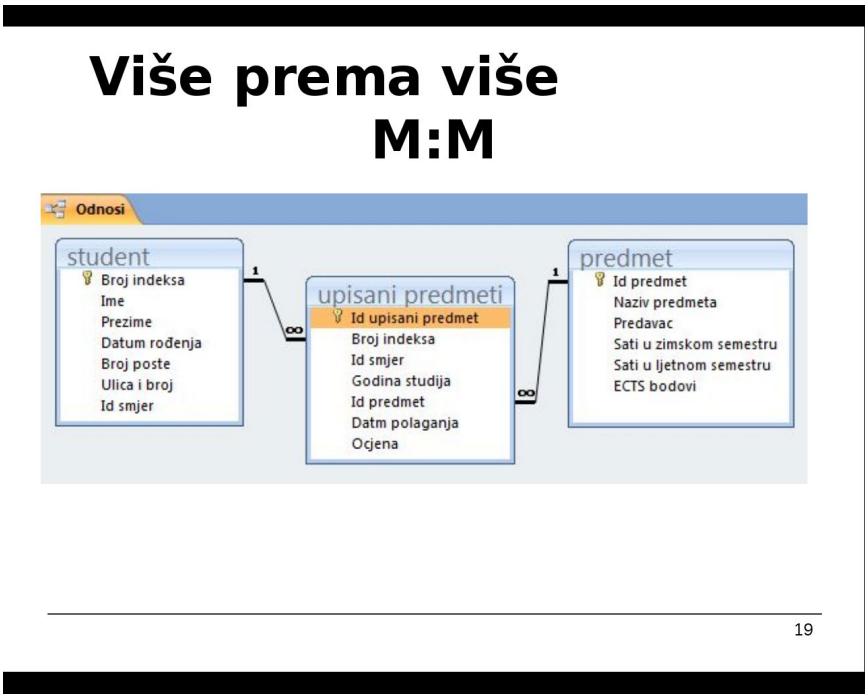


17

Jedan prema više 1:M



18



Tipovi podataka u bazama podataka:

Tekstualni tipovi podataka:

Tip podataka	**Opis**
char(n)	Znakovi fiksne duljine, maksimalno 8000 znakova
varchar(n)	Znakovi promjenjive duljine; maksimalno 8000 znakova
varchar(max)	Znakovi promjenjive duljine, maksimalno 1 073 741,824 znakova
text	Znakovi promjenjive duljine; maksimalno 2 GB tekstualnih podataka

Unicode strings:

Tip podataka	**Opis**
nchar(n)	*Unicode* podaci fiksne duljine; maksimalno 4000 znakova
nvarchar(n)	*Unicode* podaci promjenjive duljine, maksimalno 4000 znakova
nvarchar(max)	*Unicode* podaci promjenjive duljine, maksimalno 536 870 912 znakova
ntext	*Unicode* podaci promjenjive duljine; maksimalno 2 GB tekstualnih podataka

Binarni tipovi

Tip podataka	Opis
bit	Dozvoljeno 0, 1 ili NULL
binary(n)	Binarni podaci fiksne duljine, maksimalno 8000 B
varbinary(n)	Binarni podaci promjenjive duljine; maksimalno 8000 B
varbinary(max)	Binarni podaci promjenjive duljine; maksimum 2 GB
image	Binarni podaci promjenjive duljine; maksimum 2 GB

Numerički tipovi:

Tip podataka	Opis
tinyint	Dozvoljeni cijeli brojevi od 0 do 255
smallint	Dozvoljeni cijeli brojevi od -32 768 do 32 767
int	Dozvoljeni cijeli brojevi od -2 147 483 648 do 2 147 483 647
bigint	Dozvoljeni cijeli brojevi od -9 223 372 036 854,775 808 do 9 223 372 036 854 775 807
decimal(p,s)	Fiksni precizni i brojevi razmjera Dozvoljeno od -10^38 +1 do 10^38 –1. P-parametar određuje maksimum broja znamenki koji je moguće pohraniti (i s lijeve i s desne strane oznake decimale). P mora biti vrijednost od 1 do 38. Zadana vrijednost je 18. Parametar S određuje maksimalan broj znamenki pohranjen s desne strane oznake decimale. S mora biti vrijednosti od 1 do P. Zadana vrijednost je 0.
smallmoney	Valutne vrijednosti od -214 748,3648 do 214 748,3647
money	Valutne vrijednosti od -922 337 203 685 477,5808 do 922 337,203 685 477,5807
float(n)	Plutajući precizni brojevi podataka -1,79E + 308 do 1,79E + 308. Parametar *n* predstavlja bi li polje trebalo sadržavati 4 ili 8 B. *Float*(24) sadrži 4-bajtno polje i *float*(53) sadrži 8-bajtno polje. Zadana vrijednost *n*-a je 53.
real	Plutajući precizni brojevi podataka -3,40E + 38 do 3,40E + 38

Datumski i vremenski tipovi podataka:

Tip podataka	**Opis**
datetime	Od siječnja 1. 1753. do prosinca 31. 9999. s točnošću od 3,33 milisekunde
datetime2	Od siječnja 1. 0001 do prosinca 31. 9999. s točnošću od 100 nanosekundi
smalldatetime	Od siječnja 1. 1900. do lipnja 6. 2079. s točnošću od 1 minute
date	Pohranjuje samo datume; od siječnja 1. 0001. do prosinca 31. 9999.
time	Pohranjuje samo vrijeme s točnošću od 100 nanosekundi
datetimeoffset	Isto kao i *datetime2* s dodatkom vremenske zove
timestamp	Pohranjuje jedinstveni broj koji se ažurira svaki put kad se slog kreira ili modificira. *Timestamp* vrijednost bazira se na internom satu i ne odgovara realnom vremenu. Tablica može imati samo jednu varijablu *timestamp*.

Ostali tipovi podataka:

Tip podataka	**Opis**
sql_variant	Pohranjuje do 8,000 B podataka različitih tipova, osim *text*, *ntext* i *timestamp*
uniqueidentifier	Pohranjuje globalni *unique identifier* (GUID)
xml	Pohranjuje podatke formata XML, maksimalno 2 GB
cursor	Pohranjuje referencu prema pokazivaču koji se koristi za operacije na bazi.
table	Pohranjuje *result-set* za daljnje procese.