PHP

Priručnik uz seminar

Vlatka Paunović
Siniša Tomić
Hrvatska udruga za otvorene sustave i Internet

Listopad, 2006

creative commons

COMMONS DEED

Imenovanje-Dijeli pod istim uvjetima 2.5 Hrvatska

Slobodno smijete:

- umnožavati, distribuirati i javnosti priopćavati djelo
- prerađivati djelo

Pod sljedećim uvjetima:

Imenovanje. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence.

Nekomercijalno. Ovo djelo ne smijete koristiti u komercijalne svrhe.

Dijeli pod istim uvjetima. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je identična ovoj.

- U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.
- Od svakog od tih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Prethodno ni na koji način ne utječe na zakonska ograničenja autorskog prava.

Sadržaj

Programski jezik PHP

Ciljevi ovog poglavlja

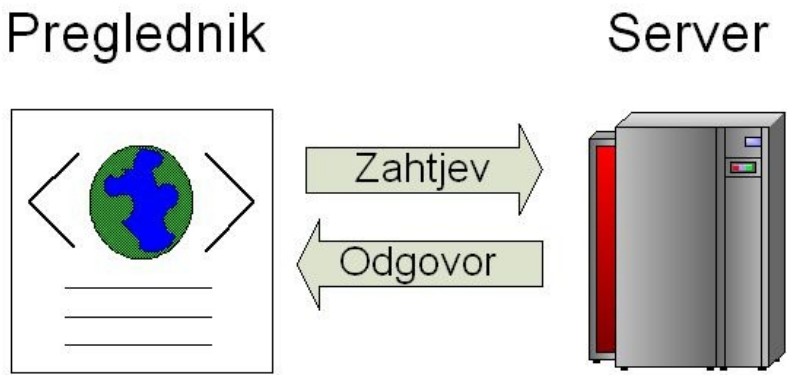
Nakon ovog poglavlja moći ćete:

- Ukratko prikazati osnovnu strukturu aplikacija temeljenih na webu
- Definirati osnovne karakteristike programskog jezika PHP
- Ukratko prikazati povijest programskog jezika PHP
- Ukratko objasniti filozofiju programa otvorenog kôda

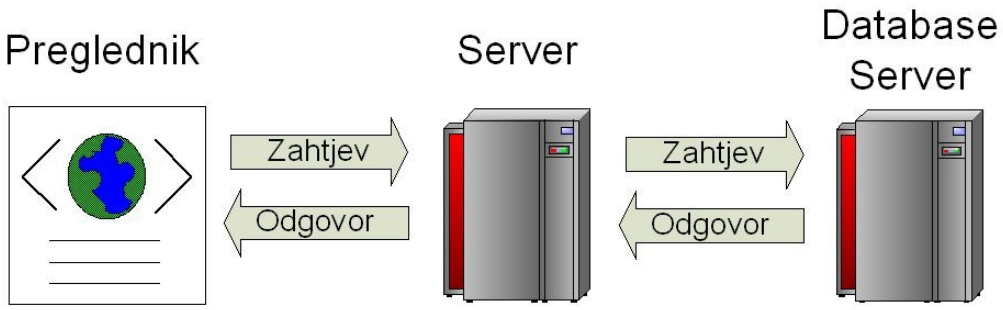
Aplikacije temeljene na webu

Arhitektura aplikacija temeljenih na webu

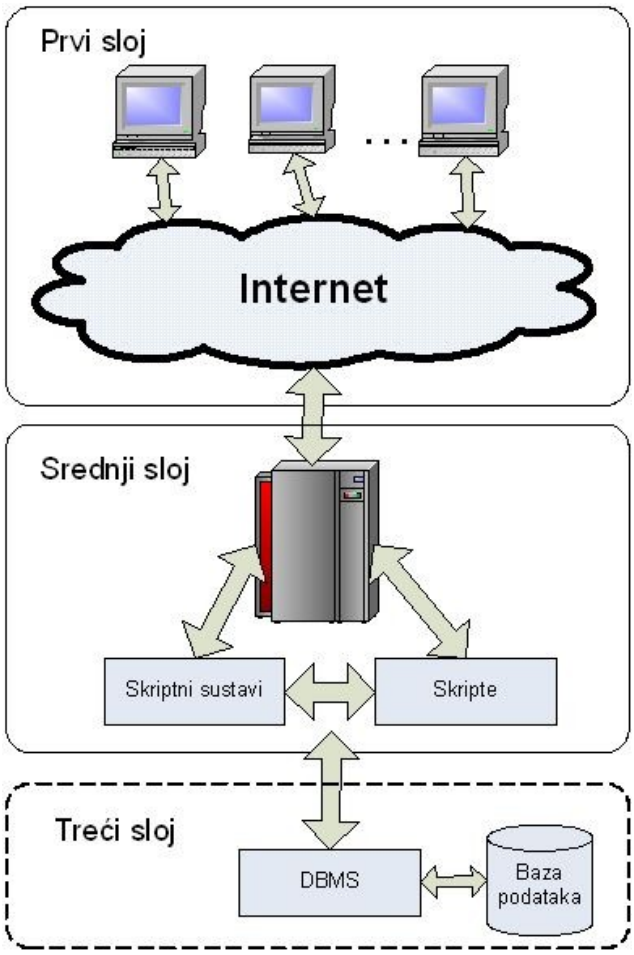
Prilikom pristupanja stranicama weba preglednik šalje zahtjev poslužitelju koji mu isporučuje datoteku u HTML obliku. Preglednik tu datoteku grafički oblikuje i prikazuje korisniku. U samom početku datoteke su bile fizički smještene na sam poslužitelj i adresa koju je preglednik tražio od poslužitelja ujedno je bila i njen stvarni naziv na lokalnom disku. Takav oblik smještaja, gdje su sve datoteke na poslužitelju nazivamo dvoslojna arhitektura (eng. Two-tier architecture). Struktura takve arhitekture prikazana je na slici:



Danas je puno češća troslojna arhitektura. U njoj datoteke i podaci nisu smješteni izravno na poslužitelju već u odvojenoj bazi podataka. Baza podataka je u pravilu smještena na odvojenom poslužitelju, a u slučaju velikog broja upita moguće je da postoji velik broj poslužitelja koji se zajedno koriste kao jedna baza podataka. Primjer takvog upita prikazan je na slici:



Kada se govori o programskom jeziku PHP on je u pravilu dio troslojne arhitekture budući da u većini slučajeva dio podataka pohranjuje se u bazu podataka. Detaljni prikaz strukture troslojne arhitekture prikazan je na ovoj slici:



Troslojna arhitektura

Troslojnu arhitekturu čine, kao što joj i ime govori, tri osnovna sloja. Prvi sloj je klijentski koji uključuje preglednik i samu mrežu Internet. Srednji sloj je poslužitelj weba na kojem se izvršavaju skriptni jezici ili izvršne datoteke, dok je posljednji, treći sloj, onaj u kojem se nalazi sustav za upravljanje bazom podataka i sama baza podataka.

Prvi sloj – preglednik i mreža Internet

Kada preglednik pošalje zahtjev za nekom stranicom poslužitelju weba, tada poslužitelj prvo izvršava program koji dohvaća podatke iz baze podataka. Dohvaćeni podaci se obrađuje, te se u HTML obliku šalju natrag pregledniku. Danas su poznatiji preglednici Internet Explorer, Mozila Firefox i Opera i putem njih korisnici pregledavaju dobivene web stranice.

Drugi sloj – poslužitelj weba i skripti jezici

Temelj srednjeg sloja je poslužitelj weba. Danas se najčešće koristi poslužitelj weba Apache, ali je vrlo čest i Internet Information Server koji je komercijalni poslužitelj. Osim ova dva, koja čine najveći dio svih poslužitelja, na raspolaganju stoji i vrlo velik broj drugih koji su u pravilu specijalizirani za neko područje – na primjer: jednostavnost upotrebe, brzina, mala upotreba memorijskih resursa i slično. Iako i ostali mogu biti dobar odabir ako se zna što je točno potrebno odabir prva dva navedena je u pravilu dobar budući su dovoljno fleksibilni i omogućavaju jednostavna proširenja.

Osim poslužitelja weba, u srednjem sloju danas se najčešće koriste skriptni jezici zbog svoje fleksibilnosti i prenosivosti na različite platforme. Jedan od popularnijih skriptnih jezika je PHP. Brojni su razlozi velike prihvaćenosti tog programskog jezika, a samo neki od najčešćih su:

- Upotreba otvorenog kôda – PHP programski jezik ima u potpunosti otvoren kôd
- Vrlo visoka razina integracije s HTML-om i potrebama Weba danas
- Mogućnost upotrebe i u najsloženijim sustavima
- Vrlo dobra prenosivost – može se bez ikakvih preinaka koristiti i pod Linux, Windows, Solaris i ostalim platformama
- Brzo izvršavanje – uz sve danas raspoložive optimizacije programi pisani u programskom jeziku PHP mogu se vrlo brzo izvršavati
- Dobra podrška zajednice – na razvoju i upotrebi programskog jezika PHP radi vrlo velik broj pojedinaca čime se osigurava njegova budućnost i ažurnost

Treći sloj – baza podataka

Sloj baze podataka se koristi za pohranu i dohvat podataka. Također, ovaj sloj omogućava istovremeni pristup podacima s nekoliko odvojenih poslužitelja, osigurava

sigurnost, tajnost i integritet podataka, a također uz odgovarajuću softversku i hardversku podršku moguće je jednostavno raditi backup svih podataka.

Zadaću upravljanja podacima u bazi podataka preuzima sustav za upravljanje bazom podataka (eng. database management system – DBMS). Sustav za upravljanje bazom podataka brine o smještaju i dohvatu podataka. Iako je sam po sebi vrlo složen on u pravilu znatno pojednostavnjuje sve operacije. Danas gotovo svi moderni sustavi za upravljanje bazom podatka za dohvat, analizu i obradu koriste jezik SQL (eng. Structured Query Language). Danas na raspolaganju postoje brojni sustavi za upravljanje bazama podataka. Neki od najpoznatijih su: Oracle, DB2, MS SQL, MySQL i PostgreSQL. Od navedenih prva tri su komercijalna, dok su MySQL i PostgreSQL besplatni programi otvorenog kôda.

Danas je za upotrebu u web aplikacijama najčešća upotreba upravo MySQL sustava za upravljanje bazom podataka zbog njene jednostavnosti i dobre podrške kako pod Microsoft Windows, tako i pod GNU/Linux platformom. Osim na ovim platformama MySQL je danas podržan i na: HP UX, Solaris, Apple MacOS X platformama za različite procesore.

Odabir arhitekture

Vrlo često pitanje je kada uopće odabrati troslojnu arhitekturu odnosno upotrebljavati bazu podataka ili ne za pohranu informacija. Preporuča se koristiti bazu podataka u ovim slučajevima:

- Više različitih korisnika istovremeno trebaju pristupati podacima
- Postoji veći broj zapisa koji se trebaju ažurirati
- Odvojeni podaci su povezani nekim vezama
- Podaci se moraju pridržavati određenih pravila – na primjer broj znakova u nizu, numerička vrijednost podatka i slično
- Potrebno je obrađivati podatke
- Potrebno je brzo pretraživati velike količine podataka
- Važno je osigurati sigurnost, tajnost ili integritet podatka
- Ažuriranje, dodavanje ili brisanje je složen ili vrlo čest zadatak

Za razliku od navedenih, postoje i slučajevi u kojima je korištenje baze podataka nepotrebno:

- Postoji samo jedna vrsta podataka
- Podaci se ne pretražuju, već se samo pohranjuju
- Upravljanje podacima je vrlo jednostavno

Programski jezik PHP

PHP svoj naziv temelji na rekurzivnoj definiciji. To je kratica za PHP: Hypertext predprocessor. Iako je svoj razvoj započeo kao „hypertext predprocessor" danas je on uvelike proširio svoje mogućnosti i ta definicija se gotovo više niti ne koristi. Osim kao podrška web aplikacijama danas je PHP moguće koristiti i kao konzolnu aplikaciju, ali je jednako moguće pisati i aplikacije u PHP-u s punim grafičkim korisničkim sučeljem, upotrebljavati OpenGL biblioteke za trodimenzionalnu vizualizaciju i slično. Danas je PHP vrlo moderan programski jezik koji je najčešće u upotrebi upravo na webu.

Svoj razvoj PHP počinje u rukama programera Rasmusa Lerdorfa koji je i danas glavna osoba zadužena za njegov razvoj. PHP je u početku tek bila pomoć Rasmusu Lerdorfu za izradu vlastite web stranice tako što je s nekoliko CGI programa pisanih u programskom jeziku C zamijenio programe pisane u programskom jeziku Perl koje je do tada koristio. Inačica 1 i 2 nisu bile previše popularne, ali 3. inačica, čija konačna inačica je izašla 1998. godine privukla je velik broj poklonika ovog novog programskog jezika. 2000. godine, izlazi i 4. inačica, da bi 2004. izašla aktualna 5. inačica koja uključuje brojne novosti kao što su uključena podrška za Unicode znakove, objektno orijentirani pristup, podrška za XML, poboljšana podrška za pristupanje MySQL bazi podataka, uključenu podršku za upotrebu Web servisa kao i noviju inačicu optimizatora kôda Zend čime su postignute znatno bolje performanse samog kôda.

Jedan od razloga vrlo dobre prihvaćenosti programskog jezika PHP je i vrlo slična sintaksa programskom jeziku C. Velik broj osnovnih funkcija ima istu sintaksu. Početnicima je također vrlo dobar je nije ograničavajući kao neki drugi programski jezici budući da nema deklaracija tipova varijabli, a moguće je i istu varijablu koristiti za pohranu različitih vrsti vrijednosti. Iako ove karakteristike programski kôd čini manje osjetljivim na pogreške jednostavnije ih je i napraviti budući da prevodilac (eng. Compiler) na njih neće upozoriti.

Danas se PHP koristi na nešto manje od dvadeset milijuna različitih web stranica prema istraživanju tvrtke Netcraft.

GNU i projekti otvorenog kôda

Projekt GNU

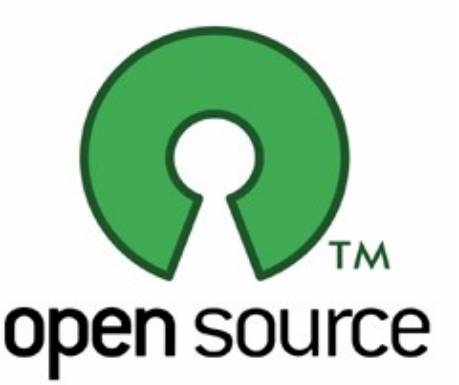
GNU projekt je započet 1984. godine, a započeo ga je Richard Stallman. Cilj projekta je od samog početka razviti slobodan operacijski sustav nalik na UNIX. GNU je rekurzivni akronim, a označava „GNU's Not Unix" odnosno „GNU Nije Unix".

Danas postoji više različitih inačica ovog operacijskog sustava, a jedna od

najpopularnijih je ona koja koristi Linux jezgru. Ona se vrlo često naziva samo Linux iako je puni naziv GNU/Linux.

Sam projekt GNU svoje prihode ostvaruje putem sponzora. Njihov najveći sponzor je organizacija FSF (Free Software Foundation) koja novac prikuplja u najvećoj mjeri od pojedinaca koji podržavaju ideju slobodnog softvera. Više od 60% novaca koji je prikupljen ostvaren je upravo donacijama pojedinaca.

Projekti otvorenog kôda – slobodni softver



Otvoreni kôd omogućava svim zainteresiranim osobama izravno sudjelovanje u unapređenju programa temeljenog na ovoj filozofiji. Sudionici projekta ne trebaju prolaziti različite razine ispitivanja, zapošljavanja ili potpisivanja ugovora o tajnosti podataka kako bi mnogi raditi na unapređenju postojećeg programa. Za sudjelovanje je dovoljno samo imati vremena, znanja i dobre volje.

Prednosti ovakvih projekata su otvorenost prema svim sudionicima i otvoreni poziv drugima na aktivno sudjelovanje. Cijeli pokret otvorenog kôda se temelji na četiri osnovne točke:

1. Sloboda pokretanja programa za sve namjene
2. Sloboda uvida u rad programa i prilagodbe potrebama – za ovo je potrebno imati pristup izvornom kôdu programa
3. Sloboda širenja kopija programa
4. Sloboda poboljšanja programa i dijeljenje te poboljšane inačice drugima

Uvod u programski jezik PHP

Ciljevi ovog poglavlja

Nakon ovog poglavlja moći ćete:

- Izraditi jednostavan program u programskom jeziku PHP
- Objasniti korištenje uvjeta i petlji
- Objasniti tipove podataka
- Raditi s nizovima znakova i poljima

Korištenje programskog jezika PHP

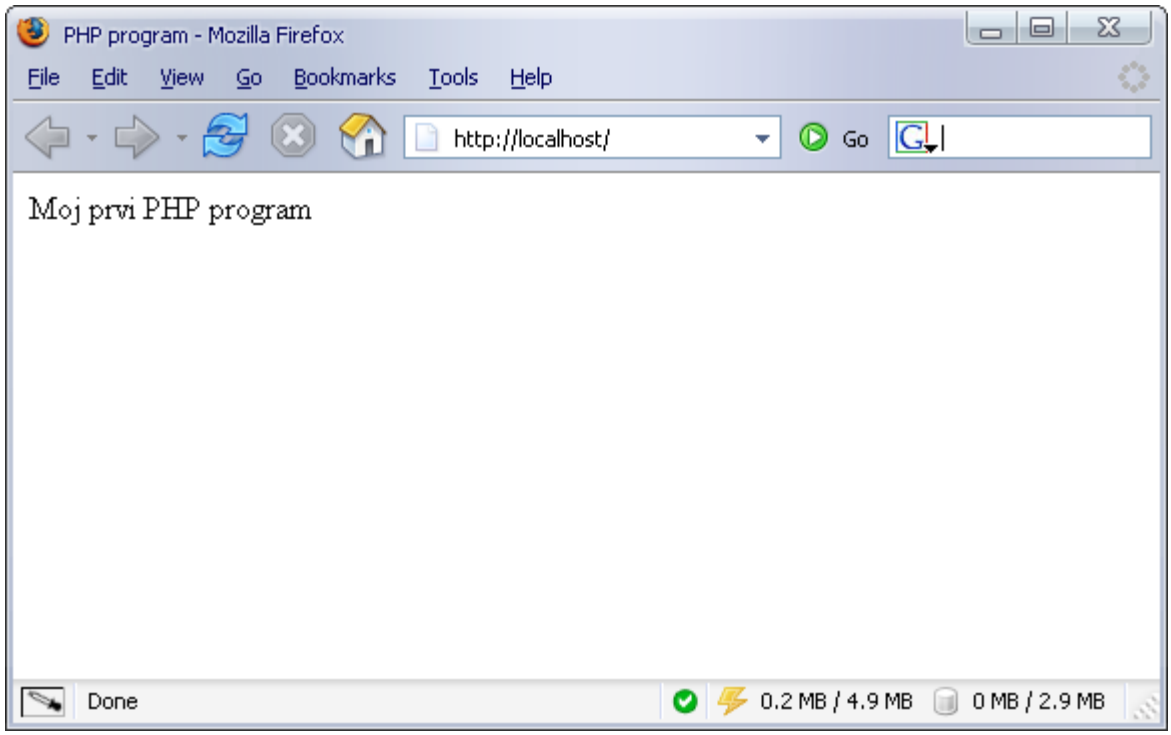
Uključivanje programa pisanog u PHP-u u web stranice

Programe pisane u programskom jeziku PHP nije potrebno prevoditi u izvršni oblik (eng. Compile) već se oni izvode prilikom pokretanja u interpreteru. Programi se pišu kao dio HTML stranice. Sam program se nalazi unutar HTML oznake koji počinje s <?PHP, a završava s ?>. Sve unutar ove oznake se smatra PHP programom i ako je na poslužitelju instaliran i ispravno podešen PHP interpreter, a datoteka završava s .php tada će se taj dio programa automatski izvršiti.

Primjer najjednostavnijeg programa pisanog u PHP-u je ispisivanje nekog teksta. Za ispisivanje teksta najjednostavnije je koristiti naredbu echo. Primjer:

```
<html>
<head>
  <title>PHP program</title>
</head>

<body bgcolor="#ffffff">
  <?php echo "Moj prvi PHP program"; ?>
</body>
</html>
```

Rezultat izvođenja je prikazan na slici:



Ako se pogleda HTML kôd koji je preglednik dobio tada se može vidjeti da je dio u kojem je bio napisan PHP kôd zamijenjen rezultatom njegovog izvođenja:

PHP program koji samo ispisuje neki tekst na ekran nije pretjerano koristan, budući da se isti učinak mogao postići jednostavnim upisivanjem kôda. Ipak, već i s ovim jednostavnim primjerom može se pokazati sljedeće:

- PHP kôd se nalazi unutar oznake <?php i ?>. Uz ovu oznaku moguće je koristiti i skraćeni oblik <? i ?> ali je za korištenje ove oznake potrebno podesiti i poslužitelj weba i PHP interpreter.
- Razmaci koji se nalaze unutar samog PHP kôda se automatski brišu. To je razlog zašto se oznaka </body> nalazi u istom retku kao i ispis programa, a ne u novom retku.

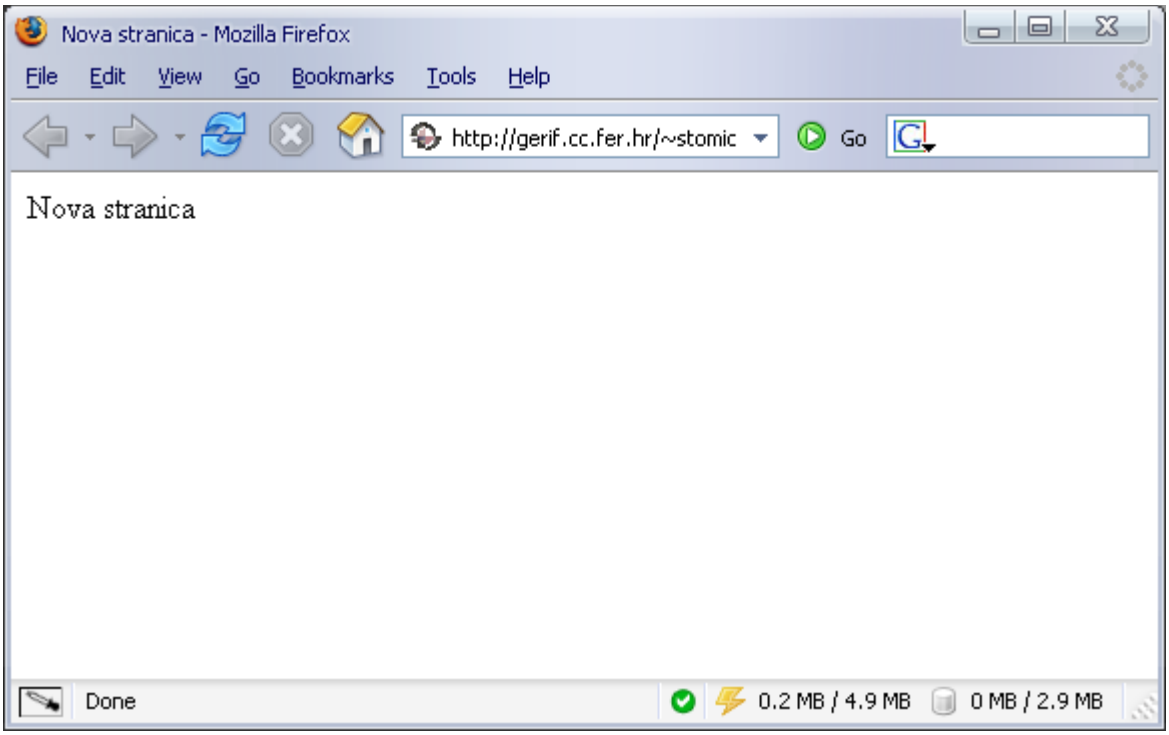
- PHP kôd je niz naredbi. Svaka naredba, kao i u programskom jeziku C mora završavati s oznakom točka-zarez (;).
- Kada se izvrši PHP kôd tada se on u ispisu zamjenjuje sa svojim rezultatom.

Zadnja karakteristika je vrlo važna jer se njome program može štititi od kopiranja, budući da osobe koje mu pristupaju ne mogu vidjeti izvorni kôd koji se izvršio, već vide samo rezultat izvršavanja. To nije karakteristika nekih drugih skriptnih jezika kao što je na primjer JavaScript čiji kôd ostaje upisan u samoj stranici.

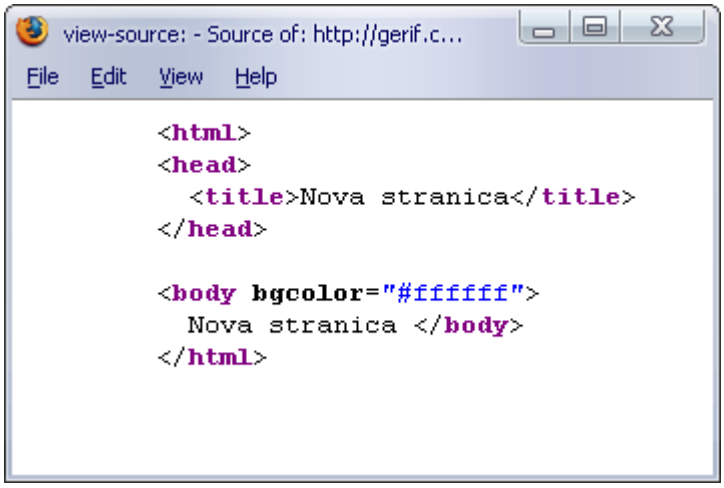
U prvom primjeru PHP kôd je napisan na samo jednom mjestu u stranici. On je mogao biti upisan i na više mjesta. Za upisivanje na više mjesta potrebno je nekoliko puta upisati oznaku <?php i ?> kako bi se uključio. Na primjer:

```
<?php $stranica = „Nova stranica”; ?>
<html>
<head>
  <title><?php echo $stranica ?></title>
</head>

<body bgcolor="#ffffff">
  <?php echo $stranica; ?>
</body>
</html>
```

U pregledniku ovaj primjer izgleda ovako:



Može se vidjeti da je i u naslovu i u sadržaju ispisan tekst „Nova stranica” iako je samo jednom definiran. Kada se pogleda HTML kôd može se vidjeti da je na dva mjesta ispisan tekst „Nova stranica”.



U ovom primjeru se može vidjeti i način određivanja vrijednosti varijable. Sve varijable u programskom jeziku PHP počinju oznakom $. Za određivanje vrijednosti koristi se oznaka jednakosti (=).

Osnove programskog jezika PHP

Korištenje komentara

Unutar samog kôda mogu se koristiti i komentari. Postoje tri oznake komentara. Prve dvije su linijski komentari, a to su oznake // i #. Iako je oznaka # dozvoljena kao komentar ona se rijetko koristi u PHP programima. Sve nakon tih oznaka se automatski smatra komentarom i kao takvo se preskače u izvođenju programa.

Primjeri linija koje su komentirane su:

```
<?php
// Neki linijski komentar
$naziv = „Novi naziv”;  // Linijski komentar - deklaracija varijable

# Drugi linijski komentar

/   / Ovo nije linijski komentar - greška
?>
```

Osim linijskih komentara postoje i blokovski komentari. Njima se može komentirati nekoliko linija kôda, a njihova oznaka je /* za početak komentara i */ za završetak komentara. Primjer blokovskog komentara dan je u primjeru:

```
/* Ovo je
      primjer
         komentara u nekoliko
            redova */

$naziv = „Novi naziv”; /*On se može koristiti i u jednoj liniji */
```

Ispisivanje podataka

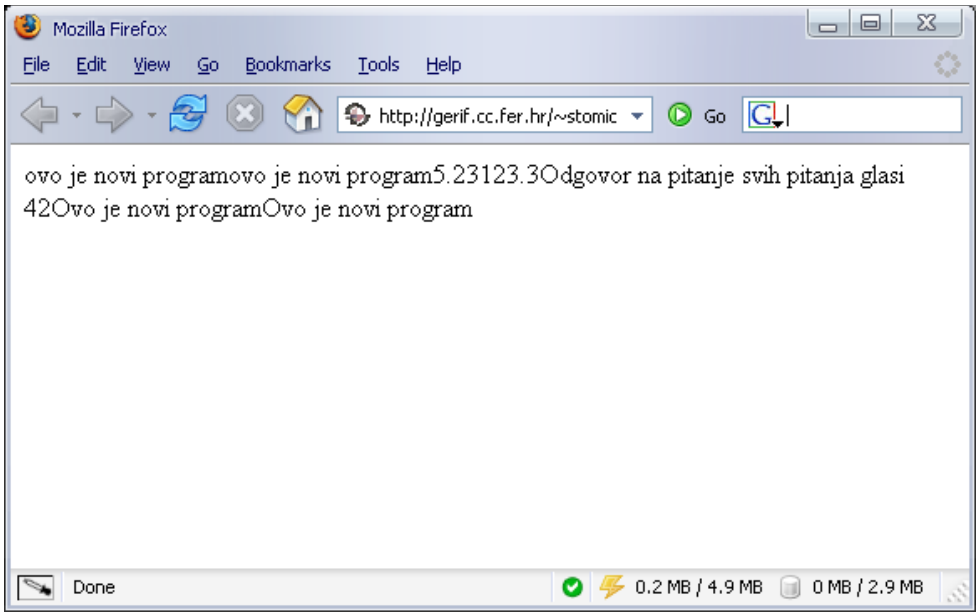
Ispis podataka ne označava kao u klasičnim programskim jezicima kao što su C ili Pascal izravno ispisivanje na zaslon korisnika. Budući da se svi programi u programskom jeziku PHP prikazuju unutar preglednika pod akcijom ispisivanja podataka smatra se da su to podaci kojima će se zamijeniti programski kôd unutar bloka <?php i ?>.

Za ispis podataka se može koristiti nekoliko naredbi. Najčešće se koriste: echo i print. Jedina razlika između ove dvije naredbe je u tome što naredba echo nema povratnu vrijednost, a print ima. U samim programima se puno češće koristi echo.

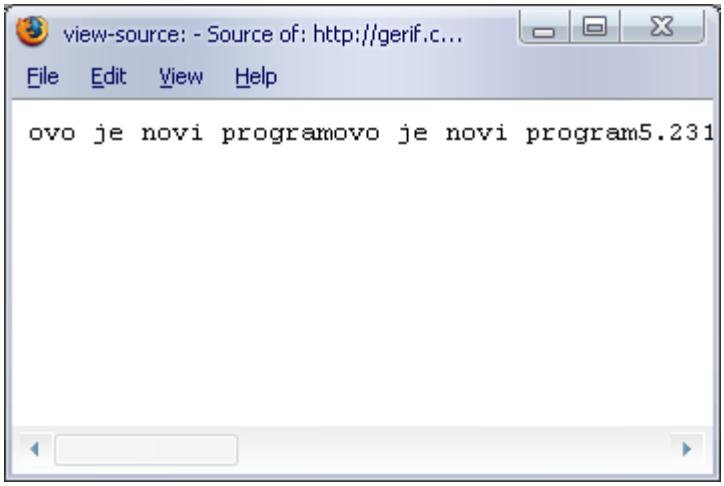
Neki od primjera korištenja su sljedeći:

```
<?php
// Primjer ispisivanja teksta s echo i print
echo „ovo je novi program”;
print „ovo je novi program”;

// Primjer ispisivanja brojeva
echo 5.23;
print 123.321;

// Primjer ispisivanja vrijednosti varijabli
$tekst = „Moj novi tekst”;
echo $text;
print $text;

// Ispis nekoliko odvojenih vrijednosti
echo „Odgovor na pitanje svih pitanja glasi „, 42;

// Obje naredbe mogu koristiti i zagrade čime izgledaju kao funkcije
echo („Ovo je novi program”);
print („Ovo je novi program”);
?>
```

U pregledniku ovaj program izgleda ovako:



Dok sam kôd izgleda ovako:



I ovdje se može vidjeti da su se svi podaci ispisali i da je php programski dio kôda zamijenjen podacima koji su se ispisali.

Ispisivanje nizova znakova

Jedan od čestih zadataka je ispisivanje nizova znakova. U programskom jeziku PHP postoje dva načina definiranja nekog niza znakova koji će se ispisati. To su označavanje niza znakova pod oznakom navodnika („), dok je drugi pod oznakom apostrofa ('). Iako se mogu koristiti obje oznake među njima postoji razlika. Osnovna razlika je u tome da li će se ispisivati vrijednost varijabli ili ne. Primjer korištenja je sljedeći:

```
<?php
$tekst = „novi”;

// Ispisuje Ovo je $tekst
echo 'Ovo je $tekst';

// Ispisuje Ovo je novi
echo „Ovo je $tekst”;
?>
```

PHP interpreter će sam zamijeniti varijablu $tekst iz primjera s njenom vrijednosti. Za određivanje naziva varijable PHP koristi jednostavan algoritam. Varijabla počinje s $, a završava s oznakom razmaka. U nekim slučajevima odmah iz vrijednosti varijable potrebno je ispisati ostatak poruke. I nije moguće upisati razmak. Takve varijable mogu se upisati unutar vitičastih zagrada.

```
<?php
$povrsina = 120;

// Ispisuje se: Površina zemljišta je 120m2.
echo „Površina zemljišta je {$povrsina}m2.“;

// Ispisuje se: Površina zemljišta je .
echo „Površina zemljišta je $povrsinam2.“;

?>
```

U prvom slučaju PHP ispravno određuje ime varijable kao $povrsina i ispravno zamjenjuje s vrijednosti 120. U drugom primjeru PHP pogrešno određuje ime varijable $povrsinam2 i budući da ne može odrediti njenu vrijednost ispisuje podrazumijevanu, a to je prazan niz znakova. Osim kod ovakvih primjera vitičaste zagrade se mogu koristiti i za ispisivanje složenih varijabli kao što su polja ili objekti.

Vrste podataka

Varijable

Varijable su nizovi znakova i brojeva koje počinju s oznakom $. Naziv varijable mora započinjati slovom ili oznakom _. Ostali znakovi mogu sadržavati brojeve, slova i oznaku _.

Sve varijable u programskom jeziku PHP razlikuju velika i mala slova. Zbog toga su sve ovo primjeri različitih varijabli: $tekst, $TEKST, $Tekst, $tekST. Prilikom upotrebe varijabli u programskom jeziku PHP nije potrebno deklarirati varijablu prije njenog korištenja kao što je to slučaj u drugim programskim jezicima. Zbog toga je pisanje programa nešto jednostavniji, ali i lakše dolazi do većih problema jer u slučaju da se pogrešno napiše ime varijable ili se pogrešno upišu velika i mala slova PHP će nastaviti izvršavati program, a kao vrijednost pogrešno napisane varijable uzimat će se njena

podrazumijevana vrijednost. Takve pogreške je teško uočiti, ali ih PHP može ispisivati podešavanjem vrijednosti postavke E_WARNING u konfiguracijskoj datoteci

Tipovi varijabli

U programskom jeziku PHP nije potrebno deklarirati varijable i na taj način zadati vrstu podataka koja se može u njih smjestiti. Osim toga, moguće je i mijenjati vrstu podatka koji se smješta u neku varijablu. PHP omogućava unutar istog bloka upisivanje numeričke vrijednosti u neku varijablu, da bi se odmah zatim u nju upisao niz znakova.

To je jedan i od problema pisanja programa, budući da je lako napisati vrlo teško čitljiv kôd koji ipak radi ispravno. Pri tome je velika vjerojatnost i pogreške budući da sam programski jezik neće upozoriti na moguće pogreške u samom programu.

Vrste podataka koje su na raspolaganju su:

- Boolean
  - Vrijednost može biti samo true (istina) ili false (laž)
- Integer
  - Vrijednost je cijeli broj
- Float
  - Vrijednost je broj s pomičnim zarezom, moguće je koristiti i eksponent: na sljedeći način: 1.23e4 == 12300 ili 5e-2 == 0.05
- String
  - Vrijednost je niz znakova

Konstante

PHP raspolože i mogućnosti korištenja konstanti. Razlika između konstanti i varijabli je u tome što se konstante ne pišu s početnom oznakom $, već se piše samo naziv konstante. Na primjer:

```
<?php
define(„PI“, 3.14);

// Ispisuje 3.14
echo PI;

?>
```

Neke konstante su u samom programskom jeziku PHP definirane. Tako nije potrebno definirati vrijednost konstante Pi, kao što je to napravljeno u primjeru, već se može koristiti već gotova definicija koja tu konstantu naziva M_PI, a ona iznosi: 3.14159265358979323846. Uz ovu na raspolaganju su i M_PI_2 (Pi/2), M_PI_4 (Pi/4), M_1_PI (1/Pi), M_2_PI (2/Pi), M_SQRTPI (sqrt(M_PI)), M_2_SQRTPI (2/sqrt(M_PI)), M_SQRT2 (sqrt(2)), M_SQRT3 (sqrt(3)), M_SQRT1_2 (1/sqrt(2)).

Izrazi i operatori

Način upotrebe izraza i operatora nije znatno drugačiji od drugih programskih jezika. Najsličniji je programskom jeziku C. Najveća razlika je u činjenici da za pisanje programa u programskom jeziku PHP nije potrebno deklarirati tipove podataka. Zbog toga nije potrebno niti koristiti „cast“ operatore.

Zbog toga, na primjer, nije potrebno prilikom dijeljenja dva cijela broja jedan od njih upotrebom „cast“ operatora pretvoriti u broj s decimalnim zarezom kako bi rezultat bio broj s decimalnim zarezom, već će to PHP sam napraviti prilikom izvršavanja programa.

Uz operatore kao što su +, -, *, / te zagrade koje se koriste za definiranje prioriteta moguće je koristiti i operatore ++ na jednak način kao i u programskom jeziku C. Na primjer:

```
<?php

// Uvećavanje vrijednosti varijable $rezultat za 1
$rezultat = $rezultat + 1;
$rezultat += 1;
$rezultat++;
++$rezultat;

?>
```

Na jednak način moguće je i umanjivanje brojeva, dijeljenje i množenje.

Kao operator za spajanje (eng. concatenation) nizova znakova koristi se operator točka (.). Tako se nizovi znakova mogu spojiti sa:

```
<?php

// Oba primjera u varijablu $tekst upisuju niz „Novi program“
$tekst = „Novi program“;
$tekst = „Novi „ . „program“;

// Rezultat je „Novi program“
$tekst = „Novi „;
$tekst .= „program“;

// Rezultat je „Novi program“
$tekst = „Novi „;
$tekst = $tekst . „program“;
?>
```

Osnovne naredbe programskog jezika PHP

Uvjeti (if i switch)

Upotreba if izraza je jednaka onoj u programskom jeziku C. Sintaksa je:

```
<?php
if ($uvjet) {
      // Blok koji se izvršava ako je $uvjet ispunjen
}
?>
```

If uvjeti se mogu kombinirati s else blokom koji će se izvršiti ako if uvjet nije ispunjen.

```
<?php
if ($uvjet) {
      // Blok koji se izvršava ako je $uvjet ispunjen
} else {
      // Blok koji se izvršava ako $uvjet NIJE ispunjen
}
?>
```

U nekim slučajevima potrebno je kombinirati nekoliko if uvjeta. To se postiže upotrebom naredbe else if:

```
<?php
if ($uvjet) {
      // Blok koji se izvršava ako je $uvjet ispunjen
}else if($uvjet2) {
      // Blok koji se izvršava ako $uvjet nije ispunjen, a $uvjet2 je
}else if($uvjet3) {
      // Blok koji se izvršava ako nisu ispunjeni $uvjet i $uvjet2,
      // a $uvjet3 jest
} else {
      // Blok koji se izvršava ako niti $uvjet niti $uvjet2 niti $uvjet 3
      // nisu ispunjeni
}
?>
```

Blok else if moguće je pisati i spojeno s elseif kao jednu naredbu. Na primjer:

```
<?php
if ($uvjet){
      // Blok koji se izvršava ako je $uvjet ispunjen
}elseif($uvjet2){
      // Blok koji se izvršava ako $uvjet nije ispunjen, a $uvjet2 je
}
?>
```

U nekim slučajevima bolje je koristiti switch naredbu. Ona djeluje jednako kao u programskom jeziku C. Na primjer:

```
<?php
switch($odabir){
  case „da“:
     // Blok koji se izvršava ako je $odabir == „da“
     break;
  case „ne“:
     // Blok koji se izvršava ako je $odabir == „ne“
     break;
  case „mozda“:
     // Blok koji se izvršava ako je $odabir == „mozda“
     break;
  default:
     // Blok koji se izvršava ako odabir nije niti jedno od navedenog.
     break;
}
```

Upotreba naredbe break je vrlo važna. Ona sprječava izvođenje naredbi koje slijede u switch bloku i nastavlja izvođenje nakon switch bloka. Ako bi se u ovom primjeru ispustila naredba break na svim mjestima tada bi se u slučaju kada varijabla $odabir ima vrijednost „da“ izvršile sve naredbe u cijelom switch bloku, dakle i naredbe pod akcijom „da“, „ne“, „mozda“ i default naredbe. Ako bi imao vrijednost „mozda“ tada bi se izvršile i naredbe u default dijelu.

Prilikom usporedbe vrijednosti varijabli važno je podatke uspoređivati s oznakom dvostruke jednakosti (==). Ova oznaka se koristi za usporedbu vrijednosti i razlikuje se od oznake jednakosti (=) koja se koristi za pridruživanje vrijednosti.

```
<?php

$tekst = 'da';

// Ispisuje: da
if($tekst == 'da'){
     echo $tekst;
}

// Ispisuje: ne
if($tekst = 'ne'){
     echo $tekst;
}
?>
```

U navedenom primjeru se prvo vrijednost varijable $tekst uspoređuje s vrijednosti „da“ i nakon što se utvrdi da je jednako ispisuje se njena vrijednost. U nastavku se varijabli $tekst pridružuje vrijednost „ne“ i ispunjava se uvjet, a ispisani tekst je „ne“.

Prilikom uspoređivanja se mogu koristiti i operatori „i“ (&&) i „ili“ (||). Na primjer:

```
<?php
if($tekst == 'da' || $tekst =='ne'){
     // Izvršava se ako je tekst 'da' ili 'ne'
}

if($tekst == 'da' && $tekst =='ne'){
     // Izvršava se ako je tekst 'da' i 'ne'
}
?>
```

Petlje (while, do ... while i for)

While, do... while i for petlje imaju jednake sintakse kao i u programskom jeziku C. Na primjer:

```
<?php
while($ispunjen_uvjet){
     // While blok
}

do{
     // Do ... while blok
}while($ispunjen_uvjet);

for($brojac = 0; $brojac < $ukupno; $brojac++){
     // For blok
}
?>
```

Prilikom korištenja petlji mogu se koristiti naredbe break i continue koje imaju jednaku ulogu kao i u programskom jeziku C.

Funkcije

U programskom jeziku PHP mogu se pisati i koristiti funkcije. U novim inačicama, kao što je inačica PHP5 mogu se koristiti i objektno orijentirane mogućnosti. Prilikom pisanja funkcija nije potrebno definirati vrstu povratne varijable za razliku od programskog jezika C. Također, ne definiraju se niti vrste ulaznih niti izlaznih varijabli u samu funkciju.

Jednostavna funkcija je oblika:

```
<?php
function naslov($tekst){
     // Tijelo funkcije
     echo „<h1>“ . $tekst . „</h1>“;
}
?>
```

Ova funkcija sadržaj niza znakova pohranjenog u varijabli $tekst ispisuje na ekran kao HTML sadržaj u obliku „<h1>tekst</h1>. Funkcija ima ime „naslov“ i jedan parametar

$tekst. Prilikom pozivanja funkcije potrebno je navesti i parametar. Funkcija vraća vrijednost upotrebom naredbe „return“ jednako kao u programskom jeziku C.

Pozivanje funkcije

Funkciju „naslov“ moguće je pozvati na ovaj način:

```
<?php
     naslov(„Ovo je naslov stranice“);
?>
```

U aktualnoj inačici programskog jezika PHP imena funkcija ne razlikuju velika i mala slova, stoga je moguće funkciju „naslov“ pozvati i na ove načine:

```
<?php
     NASLOV(„Ovo je naslov stranice“);
     Naslov(„Ovo je naslov stranice“);
     NaSlOv(„Ovo je naslov stranice“);
?>
```

U svim slučajevima pozvat će se ista funkcija – „naslov“. Zbog ove činjenice nije moguće niti dvije funkcije nazvati jednako, i razlikovati ih korištenjem malih i velikih slova u nazivu funkciju. U slučaju da se dvije funkcije pokušaju nazvati jednaku kao u sljedećem primjeru PHP interpreter će ispisati poruku o grešci:

```
<?php
function naslov($tekst){
        // Tijelo funkcije
        echo „<h1>“ . $tekst . „</h1>“;
}

function Naslov($tekst){
        // Tijelo funkcije
        echo „<h1>“ . $tekst . „</h1>“;
}

naslov("tekst");
NASLOV("test");
?>
```

Rezultat pokretanja je:

```
PHP Fatal error:  Cannot redeclare naslov() (previously declared in
index.php:2) in index.php on line 10

Fatal error: Cannot redeclare naslov() (previously declared in
index.php:2) in index.php on line 10
```

Vidljivost varijabli

Varijable imaju vidljivost jednaku kao i u programskom jeziku C. Podrazumijevana vrijednost varijable je unutar bloka u kojem je definirana. Na taj način

dvije funkcije mogu imati jednaki naziv neke varijable, ali promjena vrijednosti jedne varijable neće utjecati na promjenu vrijednosti druge varijable.

Ovakva vrsta varijabli naziva se lokalnom. Vrijednost varijable se bilježi samo unutar funkcije u kojoj se koristi. Odmah nakon što funkcija završi s izvođenjem njena vrijednost se briše. Ako je potrebno poznavati vrijednost varijable i nakon što se funkcija ponovo pozove tada je potrebno definirati statičku varijablu.

Statičke varijable bilježe svoju vrijednost unutar samo jedne funkcije, ali prilikom svakog idućeg poziva funkcije sadrže vrijednost iz prethodnog poziva te iste funkcije. Ove varijable definiraju se pomoću naredbe: static. Na primjer:

```
<?php

function naslovBrojac($tekst){
     static $brojac = 0;
     echo „<h1> Naslov „ . $brojac . „ “ . $tekst . „</h1>“;
     $brojac++;
}

naslovBrojac(„tekst“);
naslovBrojac(„tekst“);
?>
```

Funkcija iz primjera će prilikom svakog svog poziva ispisivati koji je po redu naslov i ispisat će njegovu vrijednost.

```
<h1> Naslov 0 tekst</h1><h1> Naslov 1 tekst</h1>
```

Varijablama koje su deklarirane kao statičke moguće je pristupati samo unutar funkcije u kojoj su deklarirane. Dvije odvojene funkcije mogu imati kao statičku deklarirane varijable istog imena. Te dvije varijable, iako imaju isti naziv i iako se bilježi njihova vrijednost prilikom pristupa svakoj od funkcija, su odvojene i imaju odvojene vrijednosti.

U nekim slučajevima potrebno je i koristiti varijable kojima se može pristupati iz svih funkcija i iz svih dijelova programa. Takve varijable se nazivaju globalnima. Svaka varijabla koja nije definirana unutar neke od funkcija, već je definirana unutar glavnog programa je globalna varijabla. U programskom jeziku PHP upotreba globalnih varijabli je pomalo drugačija od nekih sličnih programskih jezika. U jeziku PHP varijabla je globalna ako se unutar funkcije deklarira kao globalna, dok u sličnim programskim jezicima varijabla se treba deklarirati kao globalna unutar glavnog programa.

Za pristup globalnoj varijabli iz neke funkcije potrebno je definirati varijablu kao globalnu. To se radi upotrebom naredbe: global. Na primjer:

```
<?php
$prefiks = „Moj naslov“;

function naslovGlavni($tekst){
     global $prefiks;
     echo „<h1>“ . $prefiks . „ “ . $tekst . „</h1>“;
}

naslovGlavni(„tekst“);
naslovGlavni(„tekst“);
?>
```

Izvođenjem programa ispisuju se vrijednosti:

```
<h1>Moj naslov tekst</h1><h1>Moj naslov tekst</h1>
```

Parametri funkcije

Parametar $tekst u funkciji „naslov“ je ulazni parametar funkcije. Ulazno/izlazne parametri funkcije definiraju se upotrebom slanja reference na varijablu (eng call by reference). Parametri koje su ulazno/izlazni prije svog naziva trebaju imati upisan znak &. Parametri programa koji su ulazno izlazni će, u slučaju da im se vrijednost promjeni u samoj funkciji promijeniti vrijednost varijable kojom se funkcija pozvala.

Primjer poziva funkcije s ulaznim i s ulazno/izlaznim parametrom:

```
<?php

function naslovBrojac($tekst, &$brojac){
     $brojac++;
     $tekst = „naslov - „ . $tekst;
     echo „<h1>“ . $brojac . $tekst . „</h1>“;
}

$tekst = „Novi“;
$brojac = 1;
naslovBrojac($tekst, $brojac);
naslovBrojac($tekst, $brojac);
naslovBrojac($tekst, $brojac);
echo $tekst;
echo $brojac;
?>
```

Izvođenjem programa ispisuju se vrijednosti:

```
<h1>2naslov - Novi</h1><h1>3naslov - Novi</h1><h1>4naslov -
Novi</h1>Novi4
```

Iz rezultata izvođenja može se vidjeti da se vrijednost varijable $tekst nije promijenila i da je ostala jednaka „Novi“, dok se vrijednost varijable $brojac uvećala na 4 u glavnom programu.

U nekim slučajevima, najčešće kod proširenja postojećeg programa, moguće je

definirati parametre koji, se ne moraju upisati prilikom pozivanja funkcije. Ako se ne navedu tada će imati podrazumijevanu vrijednost, a inače onu koja se navede putem parametra. Primjer funkcije s podrazumijevanim parametrima dan je u sljedećem programu:

```
<?php
function naslovPrefiks($tekst, $prefiks = „Naslov: „){
     echo „<h1>“ . $prefiks . $tekst . „</h1>“;
}

naslovPrefiks(„Prvo poglavlje“);
naslovPrefiks(„Tehnička dokumentacija“, „Dodatak: “);
?>
```

Izvođenjem programa ispisuju se vrijednosti:

```
<h1>Naslov: Prvo poglavlje</h1><h1>Dodatak: Tehnička dokumentacija</h1>
```

Polja

Polja su vrlo važna vrsta podataka u programskom jeziku PHP. Osnovni razlog je što se većina podataka smješta u polja. Prilikom dohvata podataka iz baze podataka rezultat se smješta također u polje. Upravo iz tog razloga PHP ima vrlo razrađene funkcije za rad s poljima.

Inicijalizacija polja

Za inicijalizaciju polja može se koristiti funkcija array(). Primjer inicijalizacije polja je dan ovdje:

```
<?php
$postBrojevi = array(10000, 51000, 21000, 31000);
$gradovi = array(„Zagreb“, „Rijeka“, „Split“, „Osijek“);
?>
```

Za dohvaćanje vrijednosti podatka u polju potrebno je u uglatoj zagradi nakon naziva varijable napisati redni broj podatka koji se želi dohvatiti. Prvi podatak ima redni broj nula, drugi ima redni broj jedan i tako dalje. Ispisivanje informacije o trećem podatku u poljima prikazano je u primjeru:

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

Izvođenjem programa ispisuju se vrijednosti:

```
21000Split
```

Osim putem funkcije array() vrijednosti polja je moguće inicijalizirati i izravnim upisivanjem na odgovarajuću lokaciju. Na primjer:

```
<?php
$novoPolje[3] = 20000;
?>
```

U navedenom primjeru varijabla $novoPolje prije nije bila korištena. PHP varijablu $novoPolje automatski pretvara u tip podataka polje i na četvrto mjesto upisuje vrijednost 20000.

Ako je podatak potrebno samo dodati u polje, a nije važno na koje mjesto tada je potrebno upisati samo uglate zagrade bez vrijednost polja u koje se upisuje. Na primjer:

```
<?php
$novoPolje[] = 30000;
?>
```

PHP će sam odrediti koja je zadnja upisana vrijednost u polje i dodat će novi podatak na kraj polja.

Ispisivanje vrijednosti polja

Vrijednost u polju mogu se ispisati upotrebom funkcija print_r() ili var_dump(). Ove funkcije mogu ispisati cijelu strukturu polja u čitljivom obliku. Funkcija var_dump() je puno bolja kada je potrebno točno vidjeti i vrste podataka koji su pohranjeni, kao i broj podataka koji su pohranjeni. Funkcija print_r() za razliku samo ispisuje vrijednosti polja.

U slučaju da se ispis želi prikazati na web stranici dobro je ispis uklopiti u HTML oznake <pre> i </pre> kako bi se ispisale onako kako su zamišljene. Primjer programa:

```
<?php
     var_dump($gradovi);
     print_r($gradovi);
?>
```

Izvođenjem programa ispisuju se vrijednosti:

```
array(4) {
  [0]=>
  string(6) "Zagreb"
  [1]=>
  string(6) "Rijeka"
  [2]=>
  string(5) "Split"
  [3]=>
  string(6) "Osijek"
}
Array
(
    [0] => Zagreb
    [1] => Rijeka
    [2] => Split
    [3] => Osijek
)
```

Imenovana polja

Uz numerička polja postoje i imenovana polja. Karakteristika imenovanih polja je da se podatak ne smješta na neko mjesto koje je označeno brojem već je označeno nekim nazivom. Upotreba takvih polja je znatno jednostavnija budući da je lakše pamtiti pojmove nego brojeve. Na taj način moguće je povezati recimo nazive gradova s njihovim poštanskim brojevima. Kod imenovanog polja potrebno je prvo navesti ime polja, zatim oznaku =>, te potom vrijednost polja. Primjer definiranja imenovanog polja dan je u programu:

```
<?php

$pbrGradovi = array(„Zagreb“ => 10000,
                    „Rijeka“ => 51000,
                    „Split“  => 21000
                   );
?>
```

Osim koristeći funkciju array() imenovana polja moguće je i izravno definirati. Na primjer:

```
<?php
$pbrGradovi['Dubrovnik'] = 20000;
?>
```

Imenovana polja mogu biti i samo vrijednosti pa je tako moguće definirati polje i na ovaj način:

```
<?php
$gradoviPbr = array(10000 => „Zagreb“,
                    51000 => „Rijeka“,
                    21000 => „Split“
                   );
?>
```

Podaci u polju

Sadržaj podatka u polju može se promijeniti jednostavnim upisivanjem nove vrijednosti na mjesto stare.

Podaci u polju se mogu brisati upotrebom funkcije unset(). Na primjer:

```
<?php
unset($gradoviPbr[21000]);
unset($pbrGradovi['Split']);
?>
```

Podaci se u polju smještaju onim redoslijedom kojim su upisani u polje neovisno o tome da li je polje imenovano ili numeričko. Podaci u polju se mogu presložiti upotrebom funkcije sort() ili odgovarajuće funkcije za poredak redoslijeda elemenata u polju.

Za razliku od nekih drugih programskih jezika PHP omogućava smještaj različitih vrsti podataka u isto polje. Tako je moguće u isto polje upisati cijele brojeve, boolean varijable i nizove znakova. Na primjer:

```
<?php
$razniPodaci = array('Zagreb', 10000, 22.0, true);
?>
```

Polja mogu biti i višedimenzionalna. Odnosno, neki od elemenata polja mogu biti također polja. Na primjer:

```
<?php
$podaciOGradu = array('Zagreb',
                      10000,
                      array('Črnomerec', 'Trešnjevka', 'Centar')
                     );
?>
```

Zbog svih ovih karakteristika vrlo često u jeziku PHP nije moguće podacima pristupati koristeći klasičnu for petlju. Umjesto nje dobro je koristiti foreach petlju. Ona prolazi kroz sve elemente polja pri čemu u svakom prolazu varijabla foreach petlje poprima vrijednost sljedećeg elementa u polju. Na primjer:

```
<?php
foreach($podaciOGradu as $podatak){
     echo 'Podatak: ';
     var_dump($podatak);
}
?>
```

Izvođenjem programa ispisuju se vrijednosti:

```
Podatak: string(6) "Zagreb"
Podatak: int(10000)
Podatak: array(3) {
  [0]=>
  string(9) "Črnomerec"
  [1]=>
  string(10) "Trešnjevka"
  [2]=>
  string(6) "Centar"
}
```

Osim samog podatka moguće je i dohvatiti njegov indeks:

```
<?php
foreach($podaciOGradu as $key => $podatak){
     echo „Podatak: $key „;
     var_dump($podatak);
}
?>
```

Izvođenjem ovog primjera ispisuje se i ključ i vrijednost:

```
Podatak: 0 string(6) "Zagreb"
Podatak: 1 int(10000)
Podatak: 2 array(3) {
  [0]=>
  string(9) "Črnomerec"
  [1]=>
  string(10) "Trešnjevka"
  [2]=>
  string(6) "Centar"
}
```

Osnovne funkcije za rad s poljima

Osim već navedenih var_dump, print_r i foreach postoji nekoliko najčešćih funkcija za rad s poljima. Iako ovo nisu sve funkcije ove se najčešće koriste:

$brojac = count($polje) – i spisuje broj elemenata u polju.

```
<?php
     $polje = array('Zagreb', 'Split', 'Rijeka');
     echo count($polje);
     // Ispisuje: 3
?>
```

$polje = array_fill(3, 6, 'tekst') – upisuje vrijednost 'tekst' u elemente polja s rednim brojevima 3, 4, 5 i 6.

```
<?php
     print_r(array_fill(3,6,'Tekst'));
     // Ispisuje: Array (
     //    [3] => Tekst  [4] => Tekst  [5] => Tekst
     //    [6] => Tekst  [7] => Tekst  [8] => Tekst }
?>
```

$polje = explode(' ', $tekst) – vrijednost varijable $tekst pretvara u polje pri čemu kao znak za odvajanje koristi ' ' (razmak)

```
<?php
     print_r(explode(' ','Ovo je primjer'));
     // Ispisuje: Array (
     //    [0] => Ovo  [1] => je  [2] => primjer }
?>
```

$tekst = implode(' ', $polje) – suprotno od funkcije explode() spaja elemente polja u jedan niz znakova koristeći zadani znak za odvajanje pojedinih elemenata.

```
<?php
     $gradovi = array('Zagreb', 'Split', 'Rijeka');
     echo implode(' ',$gradovi);
     // Ispisuje: Zagreb Split Rijeka
?>
```

$vrijednost = max($polje) – vraća najveću vrijednost u polju.

```
<?php
     $vrijednosti = array(21, 32, 12, 42, 23);
     echo max($vrijednosti);
     // Ispisuje: 42
?>
```

$vrijednost = min($polje) – vraća najmanju vrijednost u polju.

```
<?php
     $vrijednosti = array(21, 32, 12, 42, 23);
     echo min($vrijednosti);
     // Ispisuje: 12
?>
```

Datumi i vrijeme

Za dohvat broja sekundi koje su protekle od 1.1.1970. do trenutnog vremena na raspolaganju je funkcija time(). Osim za dohvata aktualnog vremena moguće je i odrediti broj sekundi proteklih od 1.1.1970. za proizvoljan datum. Za to je na raspolaganju funkcija

mktime(). Primjeri korištenja ove dvije funkcije dani su u sljedećem primjeru:

```
<?php
     echo time();
    // Ispisuje broj sekundi

    //int mktime(int hour, int minute, int second,
    //           int month, int day, int year [, int is_dst])

     echo mktime(12, 30, 0, 10, 5, 2006);
    // Ispisuje: 1160044200
?>
```

U slučaju upisivanja vrijednost koji je veći od dozvoljene PHP će automatski napraviti preljev na prvi veći podatak. Tako su ova dva podatka jednaka budući da se 13. mjesec u 2005. zapravo 1. mjesec u 2006. godini:

```
<?php
     echo mktime(12, 30, 0, 1, 5, 2006);
     echo mktime(12, 30, 0, 13, 5, 2005);
     // Oba ispisuju vrijednost: 1136460600
?>
```

Praktična funkcija je strtotime() koja omogućava pretvaranje klasičnih prikaza vremena u broj sekundi proteklih od 1.1.1970. Osim unošenja datuma u nekoliko tekstualnih oblika mogući su i relativni unosi u danima, mjesecima i slično. Na primjer:

```
<?php
     echo strtotime(„12 october 2006“);
     echo strtotime(„2006-10-12“);
     echo strtotime(„10/12/2006“);
     echo strtotime(„Thu, 12 Oct 2006 00:00:00“);
     // Svi primjeri ispisuju: 1160604000
?>
```

PHP omogućava i dohvat mikrosekundi upotrebom funkcije microtime(); Na primjer:

```
<?php
     echo microtime( );
     // Ispisuje na primjer: 0.17486800 1132245217
?>
```

Prvi dio podatka predstavlja dio u mikrosekundama, dok drugi predstavlja broj sekundi proteklih od 1.1.1970. godine.

Broj sekundi proteklih od 1.1.1970. godine moguće je pretvoriti i u datum koji se može ispisati. Za pretvaranje broja sekundi u datum koristi se funkcija date(); Na raspolaganju je velik broj formata ispisa datuma:

Oznaka formata	**Opis**	**Primjer vrijednosti**
Day	---	---
d	Dan u mjesecu 2 znameke s vodećim nulama	*01* do *31*

Oznaka formata	Opis	Primjer vrijednosti
D	Tekstualni prikaz dana, tri slova	*Mon* do *Sun*
j	Dan u mjesecu bez vodećih nula	*1* do *31*
l (malo 'L')	Puni naziv dana u tjednu	*Sunday* do *Saturday*
N	ISO-8601 numerički prikaz dana u tjednu (od PHP 5.1.0)	*1* (za ponedjeljak) do *7* (za nedjelju)
S	Engleski nastavak za dan u mjesecu, dva znaka	*st*, *nd*, *rd* ili *th*. Radi ispravno s *j*
w	Numerički prikaz dana u tjednu	*0* (za nedjelju) do *6* (za subotu)
z	Dan u godini (počevši od 0)	*0* do *365*
Week	---	---
W	ISO-8601 broj tjedna u godini, tjedni počinju u ponedjeljak (od PHP 4.1.0)	Primjer: *42* (42 drugi tjedan u godini)
Month	---	---
F	Puni tekstualni prikaz mjeseca kao što je January ili March	*January* do *December*
m	Numerički prikaz mjeseca s vodećim nulama	*01* do *12*
M	Kratki tekstualni prikaz mjeseca, tri slova	*Jan* do *Dec*
n	Numerički prikaz mjeseca, bez vodećih nula	*1* do *12*
t	Broj dana u mjesecu	*28* do *31*
Year	---	---
L	Da li je prijestupna godina	*1* ako je prijestupna, 0 inače.
o	ISO-8601 oblik godine. Ima isti oblik kao i Y, osim što ako ISO broj tjedna (W) pripada prethodnoj ili sljedećoj godini koristi se ta godina. (od PHP 5.1.0)	Primjer: *1999* ili *2003*
Y	Puni numerički prikaz godine, 4 znamenke	Primjer: *1999* ili *2003*
y	Dvoznamenkasti prikaz godine	Primjer: *99* ili *03*
Time	---	---
a	Malim slovima Ante meridiem i Post meridiem	*am* ili *pm*
A	Velikim slovima Ante meridiem i Post meridiem	*AM* ili *PM*
B	Swatch Internet vrijeme	*000* do *999*
g	12-satni oblik zapisa vremena bez vodećih nula	*1* do *12*
G	24-satni oblik zapisa vremena bez vodećih nula	*0* do *23*
h	12-satni oblik zapisa vremena s vodećim nulama	*01* do *12*
H	24-satni oblik zapisa vremena s vodećim nulama	*00* do *23*
i	Minute s vodećim nulama	*00* do *59*
s	Sekunde s vodećim nulama	*00* do *59*
Timezone	---	---
e	Identifikator vremenske zone (od PHP 5.1.0)	Primjeri: *UTC*, *GMT*, *Atlantic/Azores*

Oznaka formata	**Opis**	**Primjer vrijednosti**
I (veliko i)	Da li je ili nije datum u ljetnom računanju vremena	*1* ako je ljetno računanje, *0* inače.
O	Raznika do Greenwich vremena (GMT) u satima	Primjer: *+0200*
P	Razlika do Greenwich vremena (GMT) s dvotočkom između sati i minuta (od PHP 5.1.3)	Primjer: *+02:00*
T	Vremenska zona lokalnog računala	Primjer: *EST*, *MDT* ...
Z	Pomak vremenske zone u sekundama. Pomak za vremenske zove zapadno od UTC je uvijek negativan, a za one istočno je uvijek pozitivan	*-43200* do *50400*
Full Date/Time	---	---
c	ISO 8601 datum (od PHP 5)	2004-02-12T15:19:21+00:00
r	RFC 2822 oblikovan datum	Primjer: *Thu, 21 Dec 2000 16:01:07 +0200*
U	Sekunde od Unix epohe (1. Siječanja 1970 00:00:00 GMT)	Vidi funkciju time()

Nazivi dana u tjednu i mjeseca ovise o lokalnim postavkama na računalu. Ako su podaci ispravno prevedeni na lokalnom računalu PHP će ih ispisivati na, na primjer, Hrvatskom jeziku.

Podaci o datumu mogu se provjeriti upotrebom funkcije checkdate(mjesec, dan godina) koja vraća true ako je datum ispravno upisan, a false ako nije.

MySQL – Baza podataka

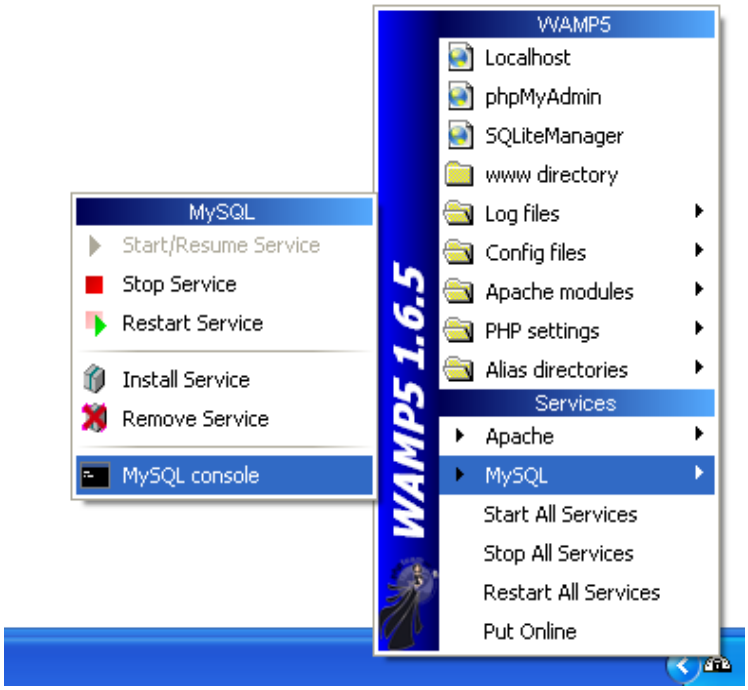
Ciljevi ovog poglavlja

Nakon ovog poglavlja:

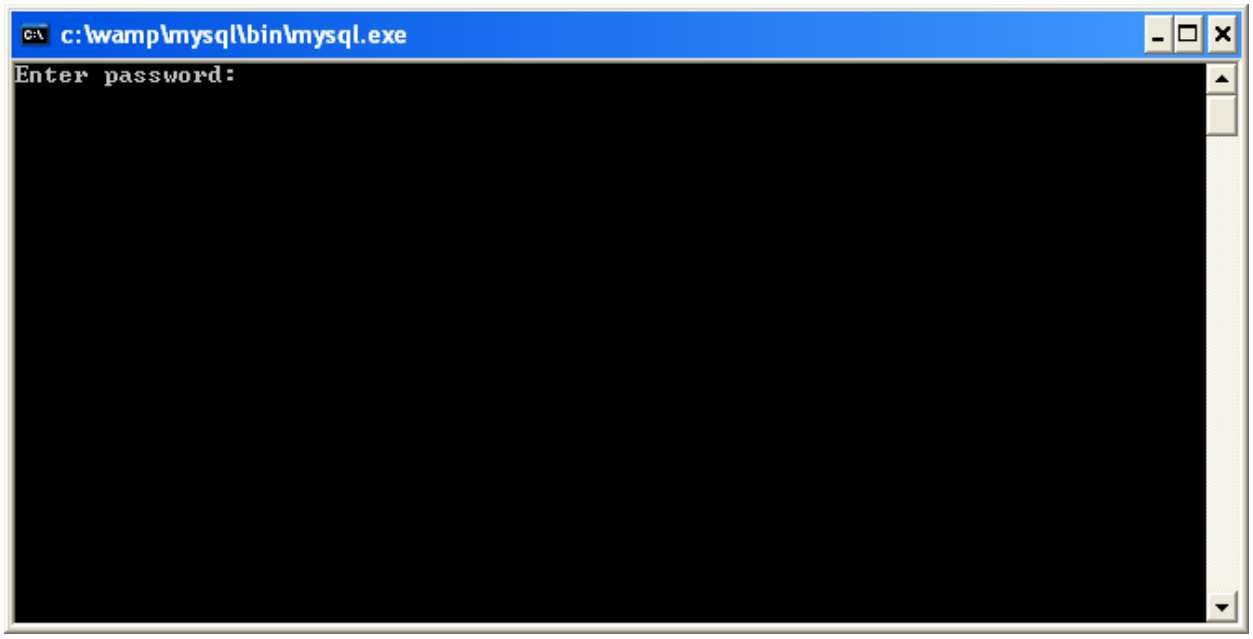
- Pokrenuti sučelje za rad s bazom podataka
- Naučit ćete osnovne pojmove o bazama podataka
- Moći ćete pokrenuti sučelje za unos podataka u bazu podataka
- Moći ćete samostalno napisati i izvršiti osnovne SQL naredbe

Pokretanje sučelja za rad s bazom podataka

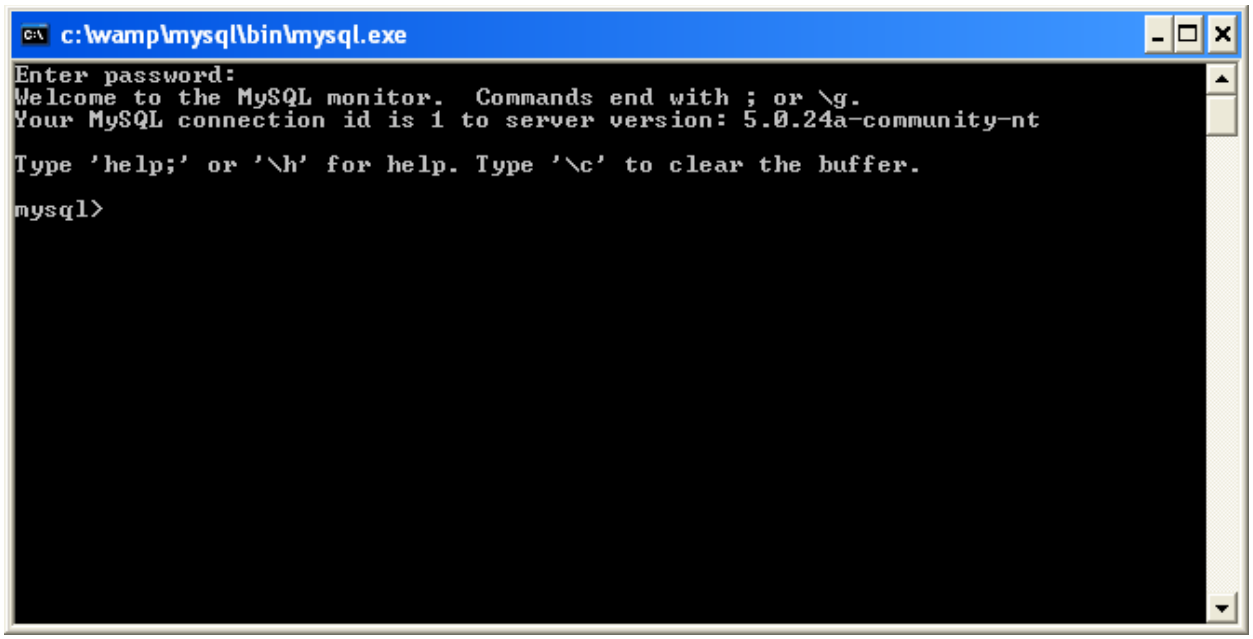
U programskom paketu WAMP, čija je instalacija opisana u ovom priručniku u poglavlju „Instalacija“, vrlo je jednostavno doći do sučelja za rad s bazom podataka.



Lijevim klikom na sličicu programa WAMP, koja se nalazi u donjem desnom dijelu „Start“ izbornika programa windows, otvorit će se WAMP izbornik. U njemu je potrebno odabrati opciju MySQL, i zatim „MySQL console“ za pokretanje sučelja.



Nakon pokretanja, potrebno je unijeti lozinku (eng. password), koja inicijalno nije postavljena, pa je samo potrebno pritisnuti tipku „Enter“. Na sustav za upravljanje bazom podataka moguće se spojiti samo s računala na kojem je on instaliran. Time je baza sigurna od stranih upada. Savjetuje se ipak postaviti lozinku.



Nakon unosa lozinke, prikazuje se sučelje sustava za upravljanje bazom podataka.

Uvod u baze podataka

Danas se baze podataka koriste, gotovo u svim web aplikacijama. U njih se mogu spremiti veće ili manje količine podataka, koje se kasnije mogu vrlo brzo i jednostavno dohvatiti. Sustav za upravljanje bazom podataka (eng. DataBase Management System - DBMS) omogućava pristup podacima kao što su unos, izmjenu i brisanje podataka, uz korištenje jezika za baze podataka (najčešće SQL).

Iako postoji velik broj različitih sustava za upravljanje bazom podatka, danas je jedna od najpopularnijih za pristup iz programskog jezika PHP MySQL.

Baza podataka se koristi za pohranu međusobno povezanih podataka. Podaci u bazi su pohranjeni na način da ne postoji nepotrebna redundancija podataka. Podaci u bazi su organizirani u tablice. Svaka baza u sebi može sadržavati jednu ili više tablica.

Tablice se koriste za spremanje skupa istih podataka. Na primjer svi podaci o gradovima se pohranjuju u jednu tablicu. Svaka tablica se sastoji od redaka, polja i atributa.

Retci su međusobno povezane vrijednosti koje su spremljeni u tablici. Atributi su stupci u tablici. Svi retci u tablici imaju iste atribute. Npr. tablica popisa Hrvatskih gradova „gradovi“ može imati atribute: ID, postanski_broj, ime_grada. Svaki atribut je određenog tipa podataka, kao što je string, integer ili date.

Relacijska baza podataka je formalni model koji koristi bazu podataka, tablice i atribute da spremi podatke i upravlja odnosima između tablica.

Osnove upotrebe SQL jezika

U većini sustava za upravljanje bazama podataka SQL jezik koji se koristi je gotovo isti, tako da je vrlo jednostavno preći s jedne na drugu bazu podataka. Razlika postoje najviše u dijelovima sintakse koje su više ili manje podržane u pojedinim sustavima za upravljanje bazama podataka.

Svaka unesena naredba završava kada se unese oznaka točka-zarez (;).

Naredbe za rad s bazom podataka

Bazu podataka potrebno je kreirati samo prvi puta unosom naredbom CREATE DATABASE**.** Time će se stvoriti nova prazna baza podataka bez tablica i podataka. Ovaj primjer će kreirati bazu pod nazivom „skolsko_natjecanje“:

```
mysql> CREATE DATABASE skolsko_natjecanje;
```

Za pristup već postojećoj bazi podatka potrebno je koristiti naredbu USE. Za pristup bazi podataka „skolsko_natjecanje“ potrebno je upisati:

```
mysql> USE skolsko_natjecanje;
```

Baza podataka se može obrisati koristeći naredbu DROP DATABASE. Važno je pripaziti prilikom brisanja baze podataka budući da se svi podaci nepovratno gube. Sintaksa naredbe DROP DATABASE jednaka je naredbi CREATE DATABASE. Na primjer:

```
mysql> DROP DATABASE skolsko_natjecanje;
```

Naredbe za rad s tablicama

Tablicu podataka potrebno je kreirati samo prvi puta, i ona će postojati u bazi podataka dok se ne obriše ili dok se ne obriše cijela baza podataka. Tablica je moguće napraviti s naredbom CREATE TABLE.

```
CREATE TABLE tableName(
     imePolja tipPodataka [PARAMETRI],
     …,
          [KLJUČEVI]
);
```

Ako se na primjer želi napraviti tablica u kojoj će biti popis svih korisnika s atributima: ime, prezime, adresa, id_grada, datum_rođenja, JMBG, potrebno je upisati:

```
CREATE TABLE korisnik (
  id serial NOT NULL,
  ime varchar(50),
  prezime varchar(50),
  adresa varchar(50),
  id_grad int(5),
  dat_rodjenja date,
  jmbg char(13),

  PRIMARY KEY (id),
  KEY (jmbg)
);
```

Tipovi podataka

MySQL podržava sljedeće osnovne tipove podataka:

- serial – cijeli broj tip podatka, koji se najčešće koristi za dodjeljivanje ID-eva. Moguće je samostalno upisati ID, ili ako se za vrijednost unese 0, sustav će automatski sam odrediti novi ID.
- int(length) - cijeli broj kojem je moguće odrediti najveću dužinu. Koristi se za određivanje ID-eva, godina, brojača i slično.
- decimal(width[,decimal_digits]) – decimalni broj, koji uključuje mogućnost određivanja broja decimalnih mjesta.
- datetime – pohranjuje datum u obliku: YYYY-MM-DD HH:MM:SS
- time – sprema vrijeme u obliku HH:MM:SS
- date – sprema datum u obliku YYYY-MM-D.D.
- timestamp - sprema datum i vrijeme u obliku: YYYYMMDDHHMMSS . Specifično za ovu vrstu podataka je da će sustav automatski upisati trenutno vrijeme ako se unese vrijednost NULL. Prilikom promjene bilo kojeg podatka u tom stupcu, sustav sam brine o promijeni vremena u trenutno.
- char(length) – niz znakova koji je uvijek zadane dužine, kao što je na primjer JMBG
- varchar(length) – za nizove znakova koji su promjenjive dužine, (kao na primjer ime osobe).
- text – niz znakova za koji se ne zna kolika će biti maksimalna dužina teksta. Na primjer: Opisni tekst, sadržaj vijesti, i slično

MySQL podržava i druge vrste podataka, koje se u svakoj verziji i nadopunjuju.

Parametri

Najčešći parametar je NOT NULL, koji označava da polje u stupcu mora imati vrijednost.

Drugi parametar, DEFAULT, omogućava automatsko dodjeljivanje podrazumijevane vrijednosti. Podrazumijevana vrijednost se automatski upisuje, ako nije navedena prilikom unosa podataka ili ako se unosi NULL vrijednost, a odabran je parametar polja NOT NULL.

Parametri se mogu istovremeno koristiti, kao u sljedećem primjeru:

```
CREATE TABLE gradovi (
   grad varchar(20) DEFAULT 'Zagreb' NOT NULL;
);
```

Ključevi

Ključ je atribut ili skup atributa čije su vrijednosti u tablici jedinstvene. U tablici može biti više ključeva, ali se jedan od ključeva uvijek proglašava primarnim. Kreiranjem primarnih ključeva baza podataka stvara indekse koji omogućavaju vrlo brzo pretraživanje i dohvat podataka.

Primjer ključeva u tablici korisnik bi bio:

```
PRIMARY KEY(id),
KEY(jmbag)
```

Umetanje, brisanje i ažuriranje podataka u tablici

Za rad s podacima u tablici, koriste se četiri osnovne naredbe: SELECT, INSERT, DELETE i UPDATE.

Umetanje podataka – INSERT

Nakon što je otvorena baza podataka, i definirane su tablice, mogu se i upisat podaci. U sljedećem primjeru dodaje se zapis za jednog korisnika u tablicu „korisnik“ s ovim atributima: „id“, „ime“, „prezime“, „adresa“, „id_grad“, „dat_rodjenja“ i „jmbg“.

```
INSERT INTO korisnik(id, ime, prezime,
                     adresa, id_grad, dat_rodjenja, jmbg)
       VALUES (0, 'Ivo', 'Ivić',
               'Maksimirska ulica 12',3,'1981-05-18','05811981330032');
```

Ova naredba će u tablici kreirati novi redak čija će vrijednost za atribut „id“ biti 1 („id“ je tipa SERIAL, a sustav započinje uvećavati brojeve od broja 1, za idući podatak „id“ će biti 2), za atribut ime će biti 'Ivo', za atribut prezime će biti 'Ivić', itd.

Ako nije potrebno ispuniti pojedini podatak za atribut, tada je moguće unijeti

vrijednost NULL, pri čemu treba pripaziti da taj atribut nema parametar NOT NULL. Primjer umetanja NULL vrijednosti:

```
INSERT INTO korisnik(id, ime, prezime,
                     adresa, id_grad, dat_rodjenja, jmbg)
       VALUES (0, 'Ivo', 'Ivić',
              NULL,NULL, '1981-05-18',NULL);
```

Za provjeru koji su sve raspoloživi atributi, i na koji način su definiranina raspolaganju je naredba: SHOW COLUMNS FROM. Na primjer za tablicu korisnik:

```
mysql> SHOW COLUMNS FROM korisnik;
```

Rezultat izvođenja naredbe je:

```
+--------------+---------------------+------+-----+---------+----------------+
| Field        | Type                | Null | Key | Default | Extra          |
+--------------+---------------------+------+-----+---------+----------------+
| id           | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| ime          | varchar(50)         | YES  |     | NULL    |                |
| prezime      | varchar(50)         | YES  |     | NULL    |                |
| adresa       | varchar(50)         | YES  |     | NULL    |                |
| id_grad      | int(5)              | YES  |     | NULL    |                |
| dat_rodjenja | date                | YES  |     | NULL    |                |
| jmbg         | char(13)            | YES  | MUL | NULL    |                |
+--------------+---------------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```

Brisanje podataka – DELETE

Brisanje jednog ili više redaka iz relacije ostvaruje se naredbom DELETE. Za brisanje svih podataka iz tablice korisnik potrebno je upisati:

```
DELETE FROM korisnik;
```

Za brisanje specifičnog retka ili redaka iz tablice potrebno je dodati WHERE dio upita. Tako, na primjer, za brisanje korisnika kojem vrijednost atributa „id“ iznosi 1 je potrebno upisati:

```
DELETE FROM korisnik
       WHERE id=1;
```

Ako je potrebno obrisati sve korisnike koji se zovu 'Ivo', tada je potrebno upisati:

```
DELETE FROM korisnik
       WHERE ime='Ivo';
```

Ako je potrebno obrisati sve korisnike čije je ime 'Ivo' i prezime počinje na slovo 'S', a rođeni su nakon 1990 godine, naredba će izgledati kao u sljedećem primjeru:

```
DELETE FROM korisnik
       WHERE ime='Ivo' AND
             prezime LIKE 'S%' AND
             dat_rodjenja > '1990';
```

U ovom primjeru upotrijebljen je operator LIKE, koji se koristi kao operator

usporedbe. Pri korištenju ovog operatora, može se koristiti:

- znak '%' - koji zamjenjuje bilo koji niz znakova
- znak '_' - koji zamjenjuje jedan znak
- znak '\' - koji ukida specijalno značenje znakova %,\ i _,a ispred kojeg je upisan.

Za brisanje svih korisnika koji se zovu 'Ivo' ili 'Marko' potrebno je upisati:

```
DELETE FROM korisnik
       WHERE ime='Ivo' OR ime='Marko';
```

Ažuriranje vrijednosti podataka – UPDATE

Izmjena jednog ili više podataka u tablici, obavlja se upitom UPDATE. Ovaj upit se najčešće koristi zajedno u kombinaciji s WHERE dijelom, koji se upotrebljava na jednak način kao i u naredbama DELETE i SELECT.

Ako se, na primjer, želi izmijeniti naziv ulice 'Ljubljanska ulica' u 'Zagrebačka ulica' potrebno je upisati:

```
UPDATE korisnik
       SET adresa='Zagrebačka ulica'
       WHERE adresa='Ljubljanska ulica';
```

Ako je potrebno izmijeniti više atributa, tada je to moguće učiniti kao u sljedećem primjeru:

```
UPDATE korisnik
       SET adresa='Zagrebačka ulica',
           id_grad=4
       WHERE adresa='Ljubljanska ulica';
```

Rezultat izvođena upita UPDATE je broj redaka koji su promijenjeni. Ako MySQL primijeti da je vrijednost već postavljena na vrijednost na koju bi ju trebalo ažurirati, podatak neće biti promijenjen i neće se uračunati u broj redaka koji su izmijenjeni.

Dohvaćanje podataka - SELECT

Naredba SELECT se koristi za dohvaćanje jednog ili više redaka iz tablice. Najjednostavniji oblik ove naredbe, dohvat svih podataka iz jedne tablice. Za dohvat svih podataka iz tablice korisnik potrebno je upisati:

```
SELECT *
   FROM korisnik;
```

U ovom primjeru se može vidjeti da je nakon riječi SELECT upisana zvjezdica. Ona označava dohvat svih stupaca iz tablice korisnik. Za dohvat samo jednog ili točno određenih stupaca, potrebno je umjesto zvjezdice upisati njihova imena i međusobno ih odvojiti zarezom.

Za dohvat svih imena i prezimena u tablici korisnik, potrebno je upisati:

```
SELECT ime,prezime
   FROM korisnik;
```

Naredba SELECT kao i naredba UPDATE, te DELETE također može sadržavati i WHERE dio. On omogućava prikaz samo onih redaka koji zadovoljavaju zadani uvjet.

Za ispisivanje samo prezimena i jedinstvenog matičnog broja osoba koje se zovu 'Ivo' potrebno je upisati:

```
SELECT prezime,jmbag
   FROM korisnik
   WHERE ime = 'Ivo';
```

U bazama podataka, najčešće postoji više desetaka ili stotina tablica, koje su u međusobnim odnosima. Ako uz tablicu korisnik postoji i tablica gradovi s atributima „id“, „postanski_broj“ i „ime_grada“, tada je moguće upisati sljedeći upit:

```
SELECT *
   FROM korisnik, gradovi
   WHERE   korisnik.id_grad=gradovi.id;
```

Ovim upitom prikazuju se svi podaci iz obje tablice, pri čemu se vrijednostima iz tablice korisnik, pridružuju vrijednosti o imenu i poštanskom broju grada.

Napredne mogućnosti programskog jezika PHP

Ciljevi ovog poglavlja

Nakon ovog poglavlja moći ćete:

- Obrađivati podatke iz obrazaca
- Koristiti $_SESSION varijablu za pohranu podataka
- Pristupati bazi podataka MySQL iz programskog jezika PHP

Obrada podataka iz obrazaca

HTML obrasci

HTML omogućava izradu web obrazaca. Web obrasci mogu sadržavati unaprijed definirane vrsta polja kao što su: kratki unos, odabir jedne od više mogućnosti, odabir više od više mogućnosti, unos kratkog teksta, unos datoteke i slično. Za svako od ovih vrsta polja postoji odgovarajuća HTML oznaka putem koje se definira vrsta polja.

Većina podataka se unosi putem <input> HTML oznake. Osnovna polja ove oznake su:

- Type – vrsta polja, a može biti:
    - Text – unos kratkog teksta – jedan redak
    - Password – unos lozinke, umjesto upisanih znakova ispisuje se zvjezdica
    - Checkbox – kvačica za odabir (da ili ne)
    - Radio – mogućnost odabira jedne od više mogućnosti
    - Submit – gumb za slanje podataka obrasca
    - Image – slika koja šalje podatke s obrasca nakon što se klikne na nju
    - Reset – postavljanje inicijalnih vrijednosti obrasca
    - Button – gumb – akcija se postiže JavaScript programom
    - Hidden – vrijednost se ne ispisuje korisniku na ekranu
    - File – odabir datoteke za slanje na poslužitelj
- Name – naziv polja – ime varijable za PHP
- Value – vrijednost polja
- Size – veličina polja – širina u znakovima

- Maxlength – najveći dozvoljeni broj znakova za upis
- Checked – da li je već označen (za radio i checbox tip polja)
- Src – URL slike (za Image tip polja)

Uz HTML oznaku input često se koristi i oznaka <TEXTAREA> i </TEXTAREA> koja se koristi za upis veće količine teksta u nekoliko redaka.

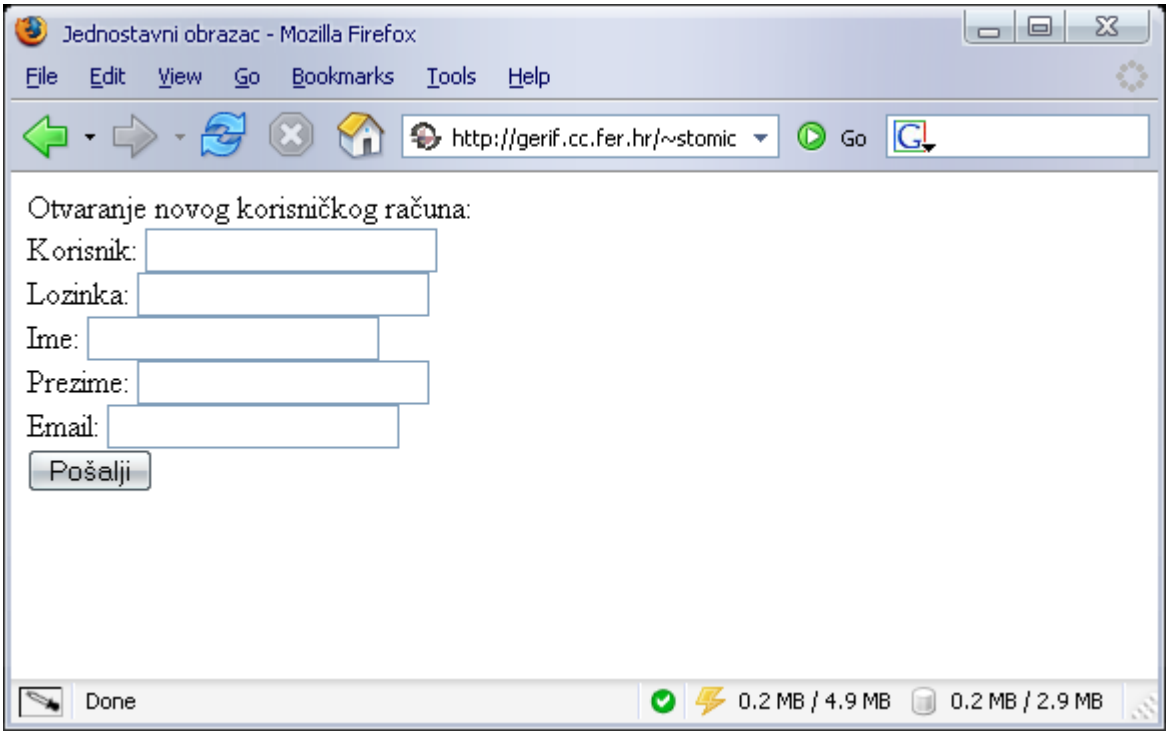
Osnovna HTML oznaka za obrazac je oznaka <FORM> i </FORM>. Osnovni parametri ove oznake su:

- Action – URL stranice na koju se šalju podaci (Na primjer: URL programa u jeziku PHP)
- Method – način slanja podataka. Može biti:
    - GET – slanje putem URL-a. Može se slati manji broj podataka, a podaci koji se šalju ne mijenjaju ništa na odredišnom računalu. Na primjer: Pretraživanje podataka
    - POST – slanje putem dodatnih parametara. Može se slati neograničena količina podataka. Podaci koji se šalju mijenjaju neke informacije na odredišnom računalu. Na primjer: Upisivanje novih podataka u bazu, mijenjanje postojećih
- Enctype – način kodiranja podataka. Može biti:
    - Application/x-www-form-urlencoded – podrazumijevani oblik
    - Multipart/form-dana – potrebno navesti ako se koristi input HTML oznaka vrste file
- Accept-charset – kodna stranica na kojoj se podaci mogu upisati u obrazac
- Accept – vrsta podataka koje poslužitelj prihvaća, ne koristi se često
- Name – naziv obrasca za dohvat polja iz JavaScript programa – danas se više ne bi trebalo koristiti, a u zamjenu bi se trebalo koristiti polje id

Za izradu jednostavnog obrasca dovoljno je samo kreirati HTML stranicu koja će sadržavati nekoliko polja definirane putem HTML oznaka <input> i smjestiti ih unutar HTML oznake <form>. Primjer obrasca putem kojeg bi se od korisnika tražilo upis podataka o korisničkom imenu, lozinci, imenu i prezimenu te e-mail adresi dan je ovdje:

```
<html>
   <head>
      <title>Prijava novog korisnika</title>
   </head>
   <body>
     Otvaranje novog korisničkog računa:<br/>
      <form action='obrada.php' method='POST'>
         Korisnik: <input type='text' name='user'><br/>
         Lozinka: <input type='password' name='pass'><br/>
         Ime: <input type='text' name='first'><br/>
         Prezime: <input type='text' name='last'><br/>
         Email: <input type='text' name='email'><br/>
         <input type='submit' value='Pošalji'><br/>
      </form>
   </body>
</html>
```

Ovaj obrazac u pregledniku izgleda ovako:



Podaci će se u primjeru prenositi putem POST parametara kao što je navedeno u parametru method budući da će se upisivati u bazu podataka čime se mijenja stanje na poslužitelju. Svi podaci šalju se PHP programu u datoteci „obrada.php“ kao što je navedeno u parametru action.

U obrascu se definiraju ova polja koja se u PHP prenose kao varijable:

- „user“ – korisnik
- „pass“ – lozinka – znakovi koji se unose bit će zamijenjeni zvjezdicama
- „first“ – ime korisnika

- „last“ – prezime korisnika
- „email“ – email adresa korisnika

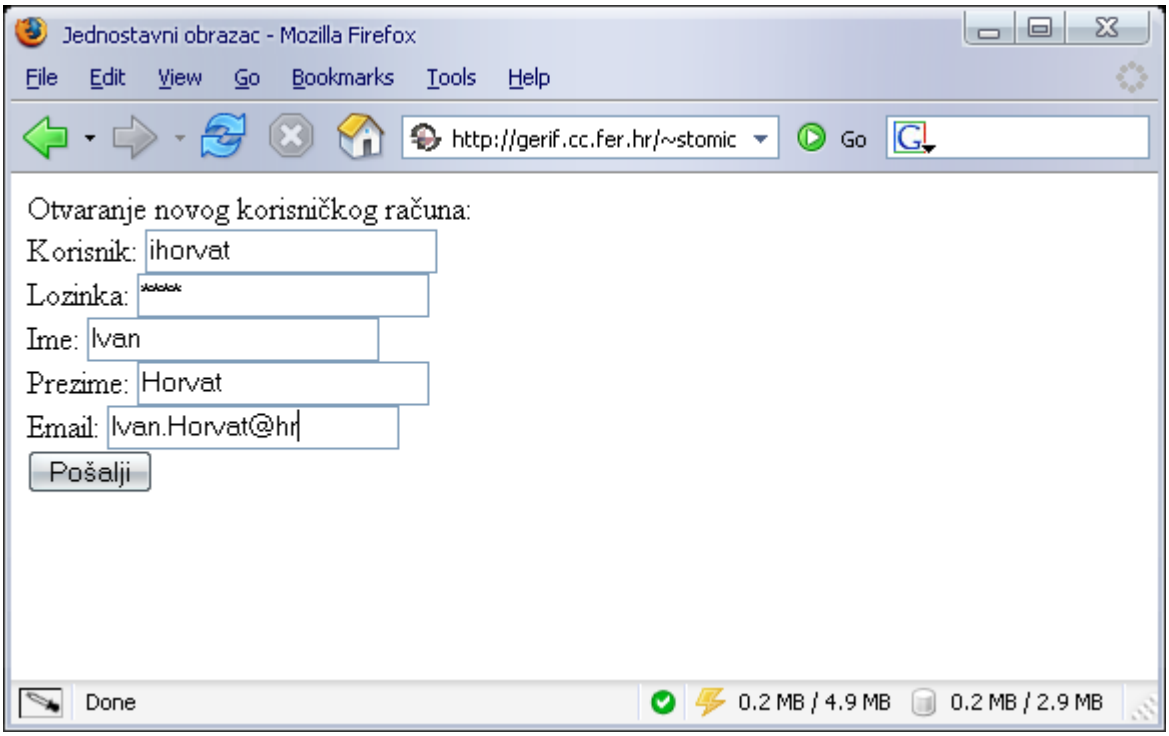
Obrada podataka iz HTML obrasca u jeziku PHP

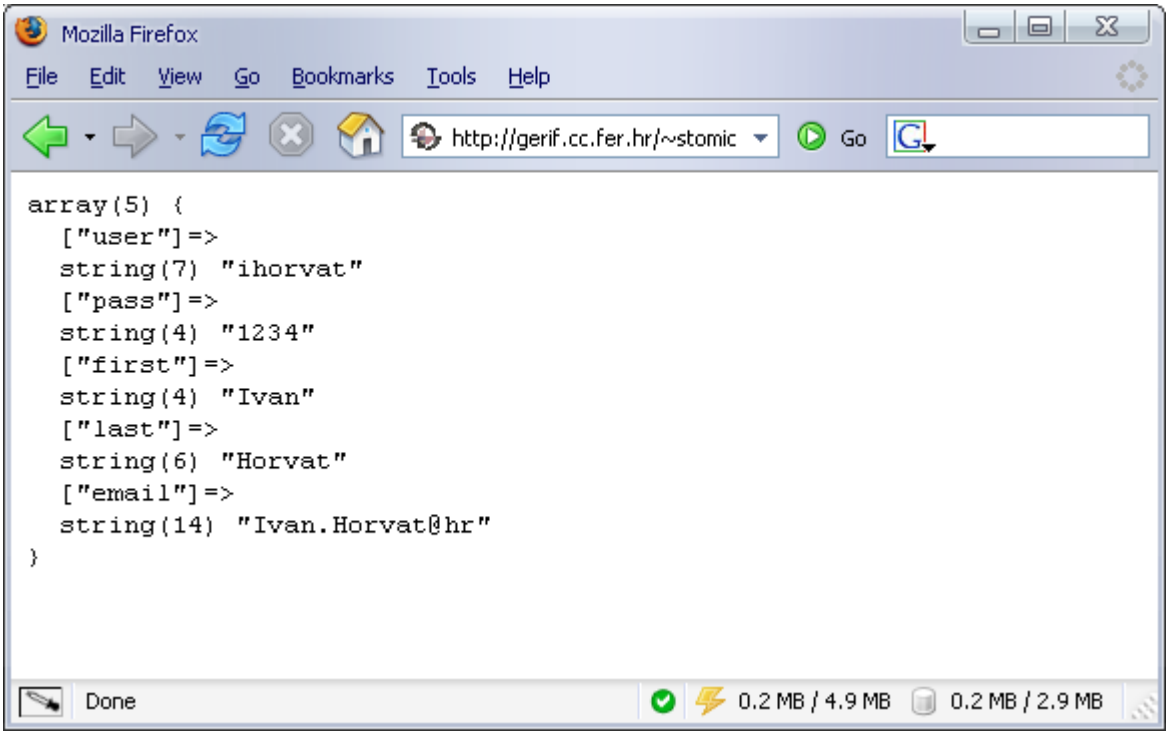
Programski jezik PHP podatke iz obrasca može dohvaćati putem globalnih varijabli $_POST i $_GET. Za razliku od ostalih varijabli ove, za pristup njihovim vrijednostima unutar funkcija, nije potrebno deklarirati kao globalne, već to PHP radi automatski. Ovisno o načinu na koji se iz obrasca podaci šalju programu dostupni su putem jednih ili drugih parametara.

Moguće je i istovremeno prenositi dio podataka putem POST, a dio putem GET parametara. Kako bi se to postiglo GET parametre je u HTML datoteci potrebno ručno navesti u action parametru HTML oznake <form>.

PHP sam obrađuje sve podatke prije nego što ih smjesti u varijable $_POST i $_GET. Podaci nakon obrade više ne sadrže kodirane znakove u obliku u kojem se prenose, već su oni već dekodirani i izravno spremni za upotrebu.

Varijable $_POST i $_GET su u stvarnosti imenovana polja. Imena u polju odgovaraju imenima varijabli u HTML obrascu, a vrijednosti su one koje su unesene u obrascu putem preglednika. Nakon ispunjavanja HTML obrasca kao na slici PHP program će dobiti ove podatke:





Sadržaj datoteke obrada.php je trenutno:

```
<pre>
<?php var_dump($_POST); ?>
</pre>
```

Osim varijabli $_POST i $_GET postoje i druge varijable kojima se može pristupiti različitim informacijama vezanim uz postavke poslužitelja ili operacijskog sustava kao što su: $_ENV i $_SERVER. Nakon dohvata vrijednosti varijabli njima se može rukovati jednako kao i bilo kojim drugim varijablama u jeziku PHP.

Prilikom korištenja varijabli unesenih od strane korisnika važno je veliku pažnju posvetiti sigurnosti. Kod PHP programa najčešća je sigurnosna greška korištenje podataka koje korisnik unese izravno bez dodatne provjere. Na taj način postoje programi pisani u jeziku PHP u kojima se unos korisnika prenosi izravno kao parametar nekom programu u operacijskom sustavu što može predstavljati veliki sigurnosni problem ako se uopće ne provjerava unos korisnika budući da korisnik može upisati takav upit kojim će se, osim željenog programa, pokrenuti i drugi, neželjeni.

Osim pozivanja programa u operacijskom sustavu čest je slučaj i izravan upis podataka u bazu podataka bez provjere. Takav upis može omogućiti vanjskom korisniku izravan pristup sustavu za upravljanje bazom podataka, a time i samom operacijskom sustavu s tek nekoliko pokušaja. Takvi upadi su vrlo teški za primijetiti, a i nakon što se otkriju ponekad je teško otkloniti njihove posljedice.

Prijava i odjava korisnika

Vrlo česta primjena programskog jezika PHP je izrada web aplikacija koje uključuju prijavu i odjavu korisnika. Pri tome je važno prijavljenim korisnicima omogućiti dodatni skup mogućnosti koji nije na raspolaganju neprijavljenim korisnicima. Za prijavu i odjavu korisnika mogu se koristiti jednostavni obrasci, a provjera ispravnosti korisničkog imena i lozinke se može raditi na nekoliko jednostavnih načina. Pitanje je kako zapamtiti da je neki korisnik prijavljen.

Prijava korisnika se u pravilu radi upotrebom Cookie varijabli koje se prenose putem preglednika. Svi današnji moderni preglednici imaju mogućnost prenošenja Cookie varijabli. Riječ je o mogućnosti da poslužitelj pregledniku pošalje vrijednost malog broja varijabli s njihovim vrijednostima. Preglednik zatim, prilikom pristupanja poslužitelju, automatski šalje te varijable s pripadnim vrijednostima na temelju čega poslužitelj može isporučiti odgovarajuće podatke.

Varijable i njihove vrijednosti se snimaju na lokalni disk s kojeg korisnik pristupa Internetu putem preglednika. Na taj način one su trajno pohranjene i nakon gašenja i paljenja računala. Osim što su pohranjene na računalu korisnika važno je i da poslužitelj zna što s njima učiniti. U samu jezgru programskog jezika PHP ugrađena je podrška za rad s tim varijablama. Budući da sam preglednik nema mogućnost pohrane velike količine podataka najčešće se podaci o korisniku pohranjuju na poslužitelj. Preglednik samo prenosi kratku identifikacijsku oznaku korisnika temeljem koje poslužitelj može rekonstruirati točno o kojoj osobi je riječ te, primjerice, koju razinu prava ta osoba ima.

Ova činjenica je vrlo važna zbog same strukture svih Web aplikacija i samog protokola dohvata Web stranica. Proces dohvata nekoliko Web stranica s jednog poslužitelja nije povezan. Tako, ako korisnik želi posjetiti neku stranicu njegov preglednik će uspostaviti vezu s poslužiteljem, poslati zahtjev za stranicom i nakon što primi odgovor prekinut će vezu. Nakon što je veza prekinuta poslužitelj gubi svaku vezu s korisnikom, a ona počinje i odmah završava svakim uspješnim učitavanjem stranice. Bez mogućnosti pohrane Cookie varijable u preglednik poslužitelj nikako ne bi mogao ponovo prepoznati istog korisnika, neovisno o tome da li je od prošle posjete prošlo nekoliko sekundi ili nekoliko godina.

Upotreba varijable $_SESSION

Iako je cijeli ovaj proces dosta složen korištenjem jezika PHP moguće ga je vrlo jednostavno koristiti. Dovoljno je samo sve podatke o korisniku koje se žele pamtiti prilikom svake posjete korisnika pohraniti u globalnu varijablu $_SESSION. Ova varijabla je imenovano polje kao što su to i varijable $_POST i $_GET. Razlika u odnosu na njih je u činjenici da ona ne sadrži podatke koje je korisnik upisao, već podatke o samom korisniku

koje aplikacija može slati sama sebi. Sadržaj varijable $_SESSION vezan je uz točno određenog korisnika i ima svoj rok trajanja. Rok trajanja ovisi o postavkama samog PHP interpretera, a obično iznosi nekoliko sati.

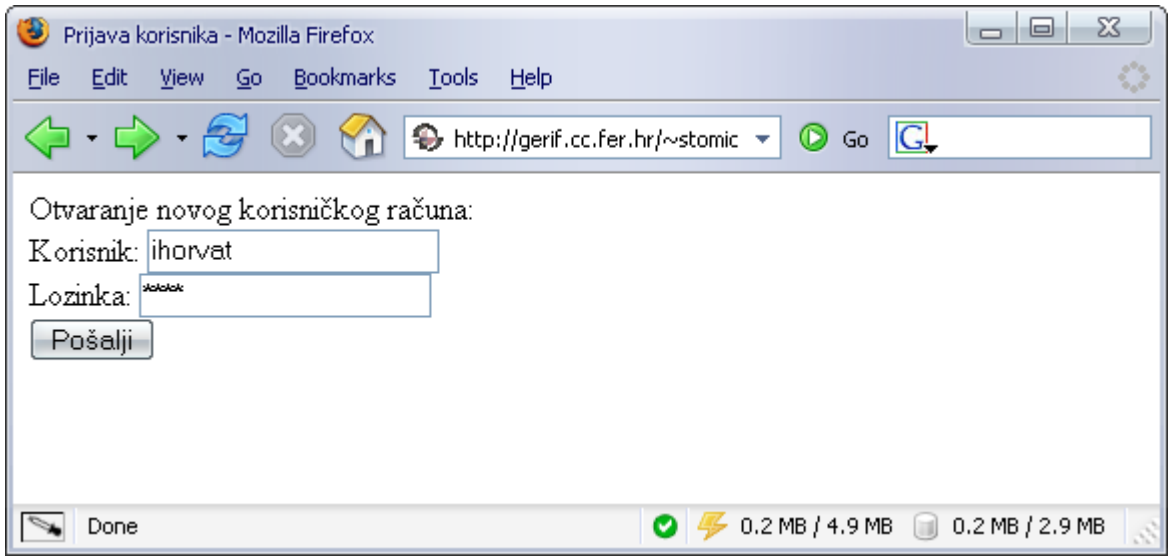
Najčešći primjer korištenja varijable $_SESSION je već spomenuta autorizacija korisnika. Ako je, na primjer, potrebno prijavljenom korisniku ispisati neku informaciju koja se ne želi prikazati neprijavljenom tada je podatke o korisniku najbolje pohranjivati u varijablu $_SESSION. Moguće je, na primjer, provjeravati vrijednost podatka login u varijabli $_SESSION. Za korištenje varijable $_SESSION važno je prvo pozvati funkciju session_start(), osim ako u postavkama PHP interpretera već nije uključen rad s tom varijablom.

```
<?php
    session_start();
    if($_SESSION['login'] == ''){
       // Korisnik nije ulogiran
    } else {
       // Korisnik je ulogiran
    }
?>
```

Za provjeru podataka o korisniku potrebno je prvo izraditi HTML obrazac za prijavu korisnika (datoteka: index.html):

```
<html>
   <head>
      <title>Prijava korisnika</title>
   </head>
   <body>
     Otvaranje novog korisničkog računa:<br/>
      <form action='login.php' method='POST'>
         Korisnik: <input type='text' name='user'><br/>
         Lozinka: <input type='password' name='pass'><br/>
         <input type='submit' value='Pošalji'><br/>
      </form>
   </body>
</html>
```

Ovaj obrazac u pregledniku izgleda kao na slici:



Nakon unosa podataka podaci se šalju datoteci login.php koja ima sljedeći sadržaj:

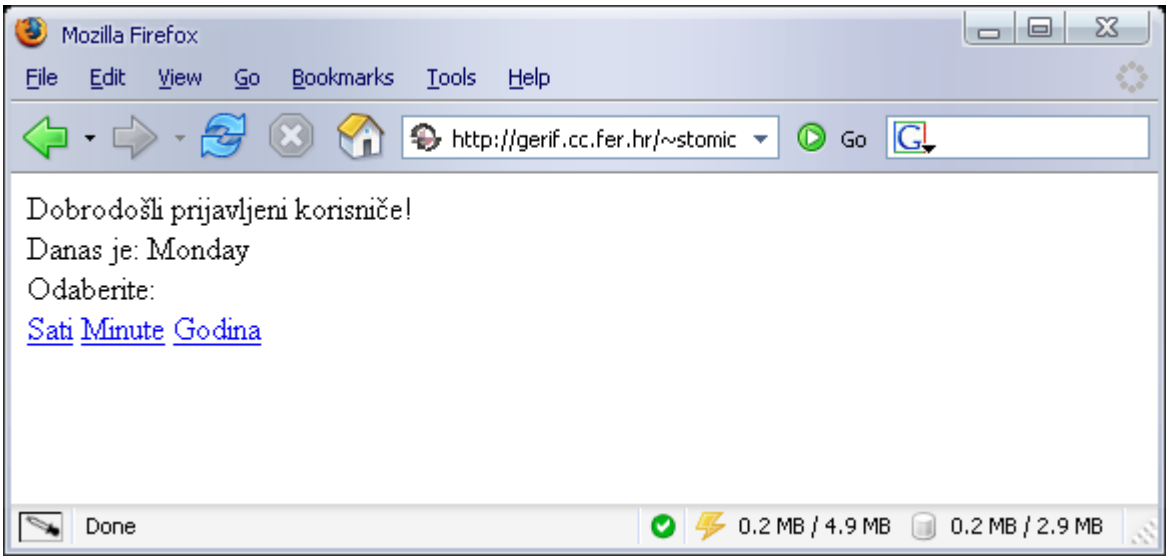
```
<?php
   // Provjera podataka
   session_start();
   if($_POST['user'] == 'ihorvat' && $_POST['pass'] == '1234'){
      $_SESSION['login'] = $_POST['user'];
   } else {
      unset($_SESSION['login']);
   }

   // Preusmjeravanje na naslovnu stranicu
   header('location: naslovnica.php');
?>
```

U provjeri podataka provjerava se da li je korisnik upisao ispravno korisničko ime (ihorvat) i lozinku (1234). Ako je postavlja se vrijednosti varijable $_SESSION['login'] na njegovo korisničko ime, a inače se briše.

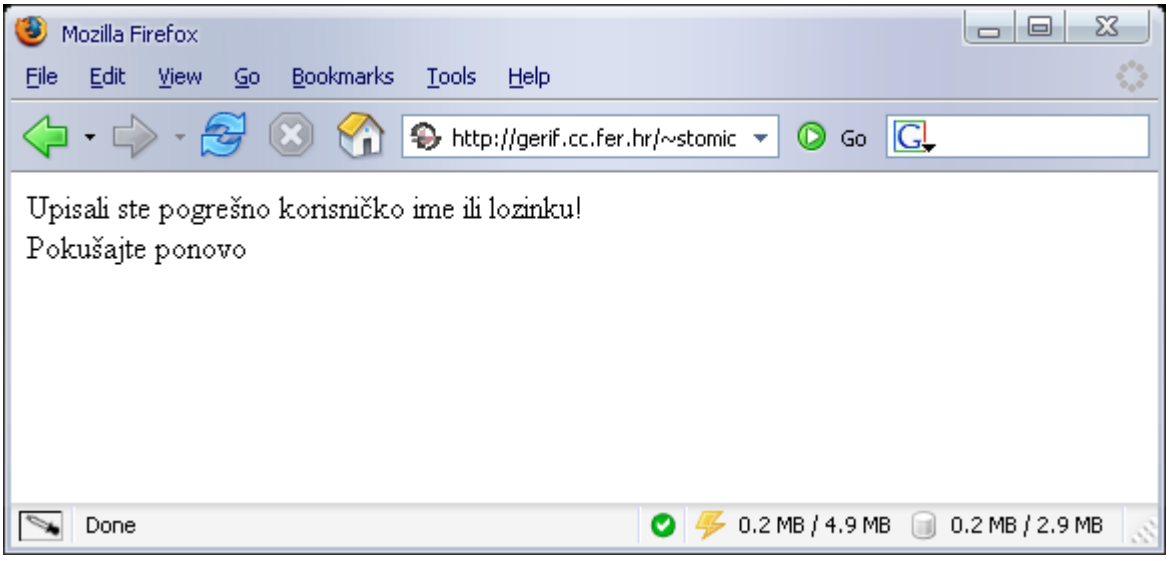
Nakon provjere koristi se funkcija header() koja preusmjerava preglednik na stranicu naslovnica.php. Sadržaj datoteke naslovnica.php:

```
<?php
      session_start();
      if(isset($_SESSION['login'])){
            echo „Dobrodošli prijavljeni korisniče!<br/>“;
            echo „Danas je: „. date(„l“). „<br/>“;
            echo „Odaberite: <br/>“;
             echo „ <a href='sati.php'>Sati</a>“;
             echo „ <a href='minute.php'>Minute</a>“;
             echo „ <a href='godine.php'>Godina</a>“;
      } else {
            echo „Upisali ste pogrešno korisničko ime ili lozinku!<br>“;
            echo „Pokušajte ponovo“;
      }
?>
```

U slučaju ispravnog unosa prijavljuje se poruka o uspješnoj prijavi.



Korisniku preglednika u slučaju pogrešnog unosa ispisuje:



Prijavljen korisnik ima linkove na datoteke sati.php, minute.php i godine.php koje

mogu imati sličan sadržaj kao i datoteka naslovnica.php.

PHP i baze podataka

Baze podataka podržane u jeziku PHP

PHP ima ugrađenu mogućnost spajanja na nekoliko različitih baza podataka. Osnovna instalacija ima podršku za spajanje na ove baze podataka:

- MySQL
- PostgreSQL
- Microsoft SQL Server
- FrontBase
- ODBC
- Sybase-CT
- Oracle (oci8)
- SQLite

U starijim inačicama jezika PHP nije moguće izravno spajanje na sve ove baze podataka, dok će u budućim popis biti vjerojatno i proširen.

Danas su najpopularnije baze podataka za web sadržaje MySQL, Microsoft SQL Server, PostgreSQL i Oracle. Spajanje s drugim vrstama baza podataka moguće je i putem ODBC veze za što je potreban odgovarajući upravljački program (eng. driver).

Svaka od ovih baza podataka ima svoje vlastite funkcije koje olakšavaju spajanje i upotrebu baze. Postoje i gotove PHP biblioteke koje same pozivaju odgovarajuće funkcije, i za pristup svim bazama podataka koriste potpuno jednake metode. Takav pristup je dobar ako se želi podržavati veći broj baza podataka, ali je njegov nedostatak sporiji pristup bazi.

Neki od najpopularnijih biblioteka su: ADOdb, PhpLib i PEAR DB. Usporavanje koje nastaje njihovom upotrebom je od 27% u slučaju ADOdb, 40% u slučaju PhpLib dok je kôd PEAR DB čak 176% u odnosu na izravni upit korištenjem ugrađenih funkcija u PHP. Ovi brojevi dobiveni su jednostavnim mjerenjem i ne moraju predstavljati točni iznos, ali barem mogu grubo prikazati zašto je bolje koristiti izravne upite ako se baza ne misli mijenjati.

Spajanje na bazu podataka MySQL

Za spajanje na bazu podatka MySQL na raspolaganju je funkcija mysql_connect(). Njeno korištenje je dano u ovom primjeru:

```
<?php
$db = mysql_connect('localhost', 'mysql_user', 'mysql_password');
?>
```

Nakon izvođenja varijabla $db sadržavat će informacije potrebne za izvršavanje upita na bazi podataka. U primjeru spaja se na sustav za upravljanje bazom podataka koji se nalazi na lokalnom računalu („localhost“), a korisnik kojim se želi spojiti je „mysql_user“. Lozinka korisnika „mysql_user“ glasi „mysql_password“.

U pravilu je dobro provjeriti da li je spajanje na sustav za upravljanje bazom podataka uspjelo ili ne. U nekim slučajevima možda sustav nije pokrenut ili je došlo do pogreške u njegovom radu. Također, nakon korištenja baze dobro ju je i zatvoriti upotrebom naredbe mysql_close(). Primjer koji pokazuje ispravni način spajanja na bazu, provjeru ispravnosti veze i prekidanje veze sa sustavom za upravljanje bazom podataka:

```
?php
$db = mysql_connect('localhost', 'mysql_user', 'mysql_password');

if (!$db) {
     die('Greška prilikom spajanja: ' . mysql_error());
}

echo 'Uspješno spojen.';
// Ostatak programa koji izvodi upit na bazu

mysql_close($db);
?>
```

Provjera ispravnosti veze se provjerava linijom: if(!$db). U slučaju da se funkcija mysql_connect nije uspjela spojiti sa sustavom za upravljanje bazom podataka ona će imati vrijednost false.

Funkcija die() prekida izvođenje ostatka programa i ispisuje poruku koja je njen parametar. Funkcija mysql_error() dohvaća poruku o pogrešci koja je nastala.

Za izvođenje upita na raspolaganju je funkcija mysql_query(). Za dohvaćanje svih informacija iz tablice korisnik nakon što je uspostavljena veza sa sustavom za upravljanje bazom podataka potrebno je upisati:

```
<?php
     $rezultat = mysql_query('SELECT * FROM korisnik', $db);
     if(!rezultat){
          die('Greška prilikom izvođenja upita '. mysql_error());
     }
?>
```

Sama funkcija mysql_query() samo izvodi naredbu, ali automatski ne dohvaća rezultate. Za dohvat rezultata koji su rezultat izvođenja ove naredbe potrebno je koristiti funkciju mysql_fetch_array().

```
<?php
while ($row = mysql_fetch_array($result)) {
   echo 'Ime: '. $row['ime'] . '#Prezime' . $row['prezime']. „\n“);
}
// Rezultat:
// Ime: Ivo#Prezime: Ivić
// Ime: Pero#Prezime: Perić
// Ime: Hrvoje#Prezime: Horvat
?>
```

Ova funkcija vraća polje koje je istovremeno i numeričko i imenovano. Odnosno, svim podacima u retku se može pristupiti i putem navođenja imena atributa, kao što je prikazano u primjeru ili putem njihovog rednog broja kojim se pojavljuje u upitu. Osim ove funkcije na raspolaganju su i funkcije mysql_fetch_row() koja vraća numeričko polje i funkcija mysql_fetch_assoc() koja vraća samo imenovano polje.

Broj zapisa u odgovoru moguće je dohvatiti upotrebom funkcije: mysql_num_fields() koja vraća broj zapisa u obliku cijelog broja. Na primjer:

```
<?php
     echo mysql_num_fields();
    // Ispisuje: 3
?>
```

Vrlo često se prilikom rada s bazom podataka traže informacije koje korisnik upisuje putem Web stranica. Na primjer pretraživanje prezimena osoba kojima je ime uneseni podatak. Čest sigurnosni propust u tom slučaju je izravno slanje upita korisnika bazi podataka. Na primjer:

```
<?php
$rezultat =
   mysql_query(„SELECT * FROM korisnik WHERE ime = '$_GET[ime]'“, $db);

if(!rezultat){
     die('Greška prilikom izvođenja upita '. mysql_error());
}
?>
```

Ako korisnik upiše kao podatak na primjer: „Pero“ izvršit će se upit:

```
SELECT * FROM korisnik WHERE ime = 'Pero';
```

Ovaj upit je potpuno u redu i dohvatit će prezime „Perić“. Sigurnosni propust

nastaje u slučaju da korisnik umjesto podataka „Pero“ upiše ovaj tekst „Pero'; DELETE FROM korisnik;“ Tada će se izvršiti ovaj upit:

```
SELECT * FROM korisnik WHERE ime = 'Pero'; DELETE FROM korisnik;';
```

Odnosno prvo će se izvršiti upit SELECT kao što je i bila želja, ali odmah potom će se izvršiti upit „DELETE from korisnik“ što nije bila želja. Ovo je vrlo čest propust kod PHP aplikacija koje izvode upit na bazi. Kako bi se spriječio korisnik u slanju ovog opasnog upita, dovoljno je podatak $_GET['ime'] poslati ugrađenoj funkciji mysql_real_escape_string() koja će iz upita obrisati potencijalno opasne znakove kao što su apostrofi i učiniti ga sigurnim za izvođenje. Također, u slučaju da se koristi mogućnost jezika PHP da sam upiše oznaku \ prije apostrofa moguće je koristiti funkciju stripslashes() koja će ih obrisati budući da će ih funkcija mysql_real_escape_string() dodati.

Primjer ispravnog programa bez sigurnosnog propusta:

```
<?php
if (get_magic_quotes_gpc()) {
   $ime = stripslashes($_GET['ime']);
}
$ime = mysql_real_escape_string($ime);

$rezultat =
   mysql_query(„SELECT * FROM korisnik WHERE ime = '$ime'“, $db);

if(!rezultat){
     die('Greška prilikom izvođenja upita '. mysql_error());
}
?>
```

Naredba mysql_real_escape_string() se odnosi samo na nizove znakova, ako upit treba sadržavati broj onda je dobro koristiti funkciju intval() koja će niz znakova uvijek pretvoriti u broj i time obrisati sve pogrešne znakove iz upisanog podatka. Na primjer:

```
<?php
$id = intval($_GET['id']);
$rezultat =
   mysql_query(„SELECT * FROM korisnik WHERE id = $id“, $db);

if(!rezultat){
     die('Greška prilikom izvođenja upita '. mysql_error());
}
?>
```

Funkcija mysql_query osim SELECT upita može izvršavati i sve ostale upite kao što su DELETE, CREATE TABLE, INSERT i ostale. U slučaju tih upita nije potrebno pozivati funkciju mysql_fetch_array() budući da ovi upiti ne vraćaju takav oblik rezultata.

Javascript

Ciljevi ovog poglavlja

Nakon ovog poglavlja:

- naučit ćete osnove JavaScripta
- naučit ćete razliku između JavaScripta i JAVE

Uvod u Javascript

JavaScript je skriptni jezik koji se koristi na webu za:

- provjeru unesenih podataka u obrascu - prije nego ih korisnik pošalje na server. Ovime se štedi procesorska snaga poslužitelja, a korisnicima se ubrzava čekanje na odgovor.
- detektiranje vrste preglednika koju koristi korisnik – kako pojedini preglednici različito prikazuju iste HTML oznake, na ovaj način je moguće korisniku prikazati dizajn koji je specifično izrađen za njegov preglednik.
- čitanje i spremanje podataka u cookies – time je moguće spremiti postavke korisnika, te prilikom iduće posjete korisnika, postaviti iste postavke
- omogućava jednostavno programiranje HTML dizajnerima – zbog svojeg jednostavnog jezika osobama koji nisu programeri omogućava umetanje jednostavnih funkcija koje dodaju dinamiku u izgledu
- dinamičko dodavanje i mijenjanje sadržaja – moguće je dodati ili promijeniti tekst ili dio HTML kôda na HTML stranicu
- reagiranje na događaje (eng. events) – izvršavanje dijela Javascript kôda, nakon pojavljivanja nekog događanja, kao što je završeno učitavanje stranice, pozicioniranje miša nad nekim HTML elementom i slično.

Sama sintaksa ovog jezika je vrlo slična jeziku C, a njegov kôd se nalazi unutar HTML kôda stranice. Zbog sličnosti imena sa programskim jezikom JAVA, većina misli da su to jednaki jezici, ali su JavaScript i JAVA različiti i po konceptu i po dizajnu. JavaScript se izvršava na klijentskom računalu, i vrlo je jednostavan programski jezik, dok je JAVA vrlo moćan i kompleksan jezik nalik na C/C++/C# koji se izvršava na samom poslužitelju.

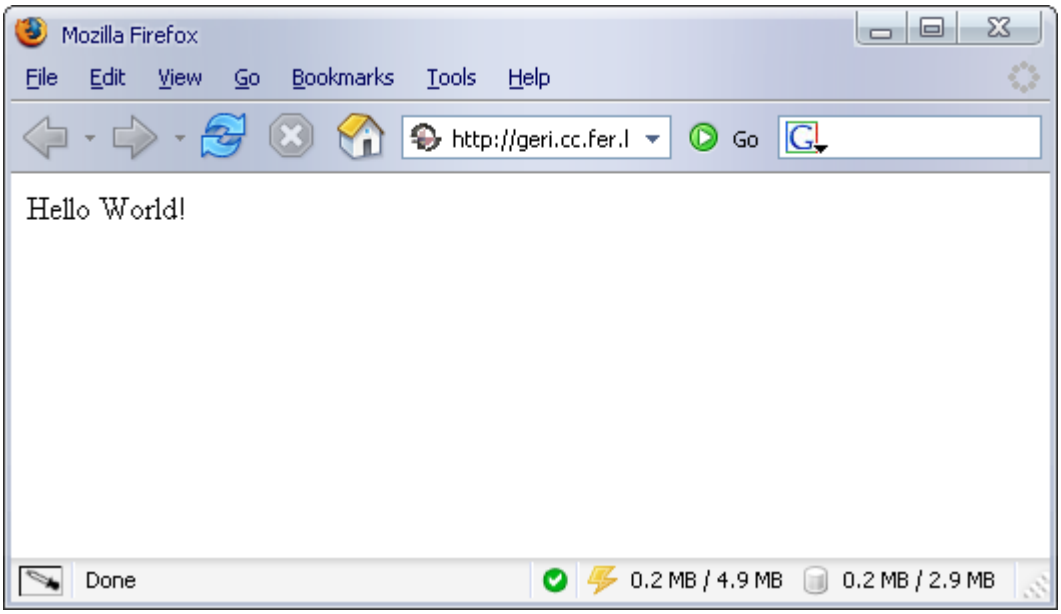
Uključivanje jezika JavaScript u web stranice

Sam kôd JavaScripta se upisuje unutar HTML stranice. Oznaka za početak JavaScripta je HTML oznaka <script language="JavaScript" type="text/javascript">, a za kraj </script>.

Primjer najjednostavnijeg programa pisanog u jeziku JavaScript je ispisivanje nekog teksta. Za ispisivanje teksta najjednostavnije je koristiti naredbu document.write(). Primjer:

```
<html>
<body>
<script language="JavaScript" type="text/javascript">
     document.write("Hello World!")
</script>
</body>
</html>
```

Rezultat izvođenja prikazan u pregledniku je:



Ako se pogleda HTML kôd u pregledniku, može se primijetiti da uključuje izvorni kôd napisan u jeziku JavaScript. To je karakteristika ovog jezika, za razliku od PHP-a kôd kojeg se u pregledniku može vidjeti samo rezultat izvršavanja, ali ne i sam kôd. Ipak JavaScript je vrlo raširen i gotovo sve web stranice ga koriste.

JavaScript je moguće umetnuti u HTML neograničeni broj puta, a razlikuje se trenutak izvršavanja ovisno o mjestu stavljanja JavaScripta unutar stranice:

- Ako je Javascript postavljen unutar BODY dijela – JavaScript se izvršava prilikom učitavanja stranice
- Ako je Javascript postavljen unutar HEAD dijela – JavaScript se izvršava

kada se pozove funkcija unutar njega. Funkcija se može pozvati iz BODY dijela ili kada se aktivira neki događaj

- Ako je JavaScript kôd u dodatnoj datoteci – JavaScript se izvršava jednako kao da se sam kôd nalazi u HEAD dijelu. JavaScript se može pozivati iz dodatne datoteke, samo je unutar HTML kôda to potrebno definirati, kako bi preglednik znao koju datoteku još treba preuzeti. JavaScript definiran na ovaj način čini HTML kôd preglednim, a može se i jednostavno dodati na više HTML stranica. Primjer JavaScripta koji se nalazi i u body i u head dijelu:

```
<html>
<head>
   <script language="JavaScript" type="text/javascript">
      ....
   </script>
</head>
<body>
   <script language="JavaScript" type="text/javascript">
      ....
   </script>
   Moja prva HTML stranica
   <script language="JavaScript" type="text/javascript">
      ....
   </script>
</body>
```

Ako se JavaScript kôd nalazi u odvojenoj datoteci, tada u datoteci nije potrebno pisati kôd unutar oznaka <script>, ali je datoteci potrebno dodijeliti nastavak .js. Primjer umetanja oznaka u HTML koje označavaju da se JavaScript kôd nalazi u datoteci „funkcije.js“ je:

```
<html>
<head>
     <script src="funkcije.js"></script>
</head>
<body>
</body>
</html>
```

Varijable

Tip varijable u jeziku JavaScript nije potrebno definirati. Imena varijabli razlikuju velika i mala slova, pa tako JavaScript ove varijable smatra različitima: Brojac, brojac, BROJAC, BROjac,… Varijable moraju započeti slovom ili oznakom podcrte '_'.

Varijabli je moguće definirati i definirati im vrijednost na sljedeće načine:

```
<script language="JavaScript" type="text/javascript">
   var brojac = 10
</script>
```

ili

```
<script language="JavaScript" type="text/javascript">
   brojac = 10
</script>
```

Varijable koje se deklariraju unutar funkcija, mogu se dohvatiti jedino iz funkcija, dok one koje su deklarirane izvan funkcija su globalne varijable i mogu se dohvatiti i unutar svih funkcija.

Kod jezika JavaScript nije neophodno upisivati oznaku točka-zarez (;) na kraju svake naredbe ako je to zadnja naredba u retku. Ako se želi navesti više naredbi u retku potrebno ih je odvojiti s oznakom točka-zarez (;).

Korištenje komentara

Unutar samog kôda mogu se koristiti i komentari. Postoje dvije oznake komentara. Prva je oznaka za linijski komentar // . Sve nakon ove oznake se smatra komentarom i kao takvo se preskače u izvođenju programa.

Primjeri linija koje su komentirane su:

```
<script language="JavaScript" type="text/javascript">
// Neki linijski komentar
 var naziv = „Novi naziv”;  // Linijski komentar – deklaracija varijable

/   / Ovo nije linijski komentar – greška
</script>
```

Osim linijskih komentara postoje i blokovski komentari. Njima se može komentirati nekoliko linija kôda, a njihova oznaka je /* za početak komentara i */ za završetak komentara. Primjer blokovskog komentara dan je u primjeru:

```
<script language="JavaScript" type="text/javascript">
/* Ovo je
      primjer
         komentara u nekoliko
            redova */

var naziv= „Novi naziv”; /*Može se koristiti i u jednoj liniji */
</script>
```

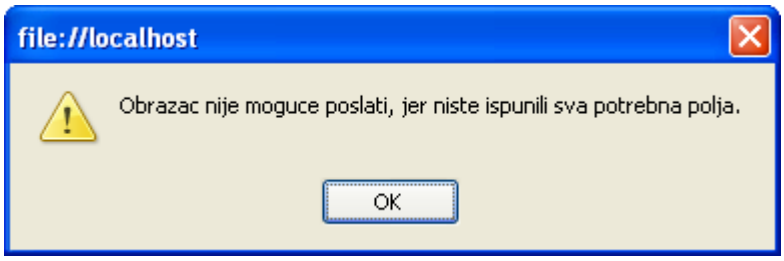
Komuniciranje sa korisnikom

Komunikaciju sa korisnikom moguće je ostvariti kroz jednu od ovih mogućnosti: prozor s upozorenjem (eng. Alert Box), potvrdni prozor (eng. Confirm Box) ili prozor za unos vrijednosti (Promt Box).

Prozor s upozorenjem se najčešće koristi kada se želi vidljivo prenijeti neka važna informacija do korisnika. Vrlo često se koristi u obrascima, kako bi se korisnika koji želi poslati ispunjeni obrazac, upozorilo da nije ispunio sva polja, te da ne može nastaviti dok ne ispuni sva obavezna polja. Ovaj prozor ispisuje poruku koja se šalje unutar funkcije, a gumb „OK“ se koristi za potvrdu da je poruka primljena. Primjer:

```
<script language="JavaScript" type="text/javascript">
alert("Obrazac nije moguce poslati, jer niste ispunili sva potrebna
polja.");
</script>
```

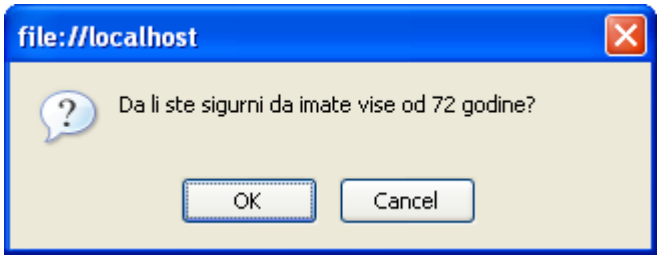
Izvršavanjem ovog dijela JavaScript kôda korisniku će se prikazati sljedeći prozor:



Potvrdni prozor se najčešće koristi kada se želi da korisnik potvrdi ili odbaci neku akciju ili informaciju. Najčešće se koristi u obrascima kako bi se provjerilo da li je korisnik siguran da želi obrisati ili dodati neki podatak. Ovaj prozor ispisuje poruku, koja je zadana unutar funkcije, te mogućnosti „OK“ i „Cancel“, kojima korisnik može vratiti potvrdni ili niječni odgovor. Primjer:

```
<script language="JavaScript" type="text/javascript">
     confirm("Da li ste sigurni da imate vise od 72 godine?");
</script>
```

Izvršavanjem ovog dijela JavaScript kôda korisniku će se prikazati sljedeći prozor:



Treća vrsta prozora je prozor za unos vrijednosti. Ova vrsta naredbe se ne koristi baš često, ali ju je moguće iskoristiti, ako je potrebno na brzinu napraviti jednostavni obrazac s jednim poljem za unos vrijednosti. Funkcija za poziv naredbe prihvaća dva

parametra. Prvi parametar je poruka koja će se ispisati, a drugi je podrazumijevana vrijednost koja će se korisniku prikazati u polju. Primjer:

```
<script language="JavaScript" type="text/javascript">
     prompt("Vase ime je:", "Ivana")
</script>
```

Uvjeti (if i switch)

Upotreba izraza if i switch jednaka je onoj koja se koristi u programskom jeziku C ili skriptnom jeziku PHP.

Petlje (while, do ... while i for)

Upotreba petlji jednaka je onoj koja se koristi u programskom jeziku C ili skriptnom jeziku PHP.

Funkcije

Upotreba funkcija jednaka je onoj koja se koristi u programskom jeziku C ili skriptnom jeziku PHP. Varijable koje se deklariraju unutar funkcija, mogu se dohvatiti jedino iz funkcija, dok one koje su deklarirane izvan funkcija su globalne varijable i mogu se dohvatiti i unutar svih funkcija.

Smarty

Ciljevi ovog poglavlja

Nakon ovog poglavlja moći ćete:

- Definirati što je to sustav predložaka Smarty
- Uključiti Smarty predloške u PHP program
- Koristiti osnovne mogućnosti Smarty predložaka

Upotreba predložaka

PHP je od početka zamišljen kao programski jezik za upotrebu u Web aplikacijama. Rezultat izvođenja programa pisanog u jeziku PHP upisuje se izravno u HTML stranicu. Vrlo često je rezultat izvođenja je potrebno vizualno prilagoditi ostatku stranice.

Upotrebom izravnog ispisivanja poruka upotrebom naredbe echo ili print to može postati vrlo teško. Budući da se dio podataka koji se ispisuje može zbog složenosti samog jezika PHP nalaziti u sklopu različitih funkcija ili čak drugih datoteka koje se mogu uključiti lako može biti nejasno što se točno ispisuje, u kojim slučajevima i na koji način. Dodatni problem nastaje kada je potrebno dizajn stranice prepustiti osobi koja ne razumije programski jezik PHP, ali dobro zna HTML, na primjer profesionalnom dizajneru.

Kako bi se olakšao ovaj problem uveden je sustav predložaka. Njihova uloga je odvojiti programski kôd od dijela za prikaz. Na taj način programeri brinu samo o programskom kôdu, odnosno „poslovnoj logici“, dok dizajneri brinu o vizualnom izgledu. Promjena jednog dijela ne mijenja drugi čime se vrlo brzo mogu napraviti različite aplikacije koje izgledaju jednako ili vrlo slično, kao i istoj aplikaciji napraviti nekoliko vizualnih identiteta.

Za programski jezik PHP postoji nekoliko različitih sustava predložaka, a najpoznatiji među njima su:

- Smarty: http://smarty.php.net/
- PHP Savant: http://phpsavant.com/yawiki/
- PHPlib: http://phplib.sourceforge.net/
- Yats: http://yats.sourceforge.net/
- FastTemplate: http://www.thewebmasters.net/php/FastTemplate.pHTML
- SimpleTemplate: http://simplet.sourceforge.net/

- Yapter: http://yapter.sourceforge.net/

Većina od ovih sustava ima svoje prednosti i nedostatke. Danas se vrlo često koristi sustav predložaka Smarty. Njihova najveća prednost je brzina, dok je prednost predložaka Smarty u velikoj mogućnosti prilagodbe i izvršavanju najsloženijih zadataka. Najveći konkurent sustavu predložaka Smarty je sustav predložaka Savant. Savant se definira upotrebom PHP naredbi, pa dizajneri ne trebaju učiti novi programski jezik, neovisno o tome koliko je jednostavan, ali mu je to ujedno i nedostatak budući da dizajneri time imaju mogućnost korištenja svih PHP naredbi čime se može dio funkcionalnosti premjestiti u predloške što nije dobro.

Sustav predložaka Smarty

Osnovne mogućnosti

Osnovne mogućnosti predložaka Smarty su:

- Promjena varijabli – eng. Variable Modifiers – omogućava jednostavnu promjenu sadržaja varijabli. Na primjer: pretvaranje teksta u velika slova, brisanje teksta nakon zadane dužine i slično
- Funkcije predložaka – eng. Template Functions – mogućnost definiranja složenih funkcija koje na temelju zadanih parametara ispisuju cijele djelove HTML stranice. Na primjer na temelju polja ispisuju HTML tablicu
- Ispravljanje pogrešaka – eng. Debugging – korištenjem konzole za praćenje pogrešaka moguće je jednostavno pronaći pogrešku i ispraviti ju
- Sustav dodataka – eng. Plug-ins – na raspolaganju je velik sustav dodataka koji mogu znatno proširiti inicijalne mogućnosti ovog sustava predložaka
- Filtri – eng. Filters – na raspolaganju je mogućnost pregledavanja cijelog kôda u filtrima koji se izvode prije i nakon izvršavanja predloška čime je moguće, na primjer, izbrisati sve HTML komentare i time smanjiti veličinu stranice ili dodati informacije ako je to neophodno

Sustav predložaka Smarty je dostupan pod GNU općom licencijom čime se besplatno njegovo korištenje u osobne, edukativne i komercijalne namjene kao i kod svih ostalih programa temeljenih na filozofiji otvorenog kôda.

Instalacija

Prvo je potrebno preuzeti aktualnu inačicu sustava predložaka Smarty s adrese: http://smarty.php.net/download.php. Aktualna inačica uvijek sadrži sve izmjene koje su uočene u prethodnima čime se osigurava sigurnost izvršavanja.

Preuzetu datoteku potrebno je otpakirati u web direktorij gdje se nalaze PHP datoteke koje će koristiti predloške Smarty. Nakon što se datoteka otpakira kreirat će se otprilike ovakva struktura mapa:

- smarty.class.php
- smarty_Compiler.class.php
- config_File.class.php
- debug.tpl
- internals/
- plug-ins/

Prve su četiri datoteke, a posljednje dvije su mape. Ove datoteke nije potrebno uređivati i sve su potrebne za ispravno izvršavanje programa. Osim ovih mapa potrebne su i ove: templates/ te templates_c/ i cache/. U mapi templates trebaju se nalaziti datoteke predložaka koje će se koristiti iz PHP programa, dok će se u mape templates_c/ i cache/ automatski pohranjivati kompilirane datoteke predložaka za brže izvršavanje. Zbog toga je potrebno poslužitelju weba omogućiti pisanje u mape templates_c/ i cache/.

Poslužitelj weba mora imati pravo čitanja svih datoteke i mapa za ispravan rad.

Uključivanje predložaka Smarty u PHP program

Kako bi se uključili predlošci Smarty u program potrebno je upisati:

```
<?php
require('Smarty.class.php');
$smarty = new Smarty;
?>
```

Ovim primjerom uključuje se datoteka Smarty.class.php koja je osnovna datoteka predložaka Smarty i kreira se novi objekt pod nazivom $smarty koji je tipa Smarty.

Smarty se koristi za vizualizaciju podataka koji se kreiraju u jeziku PHP. Zbog toga je najvažnija metoda varijable $smarty ona za definiranje vrijednosti neke varijable. Kako bi se na primjer definirana varijabla 'ime' u predlošku Smarty potrebno je koristiti metodu assign(). Na primjer:

```
<?php
require('Smarty.class.php');
$smarty = new Smarty;

$smary->assign('ime', 'Moje ime');
?>
```

Ovim primjerom je definirana varijablu 'ime' u predlošku Smarty sa zadanom vrijednosti 'Moje ime'.

Osim zadavanja varijabli, potrebno je i zadati naziv datoteke predloška koji će se ispisati. Naziv datoteke se zadaje upotrebom metode display(). Na primjer:

```
<?php
require('Smarty.class.php');
$smarty = new Smarty;

$smary->assign('ime', 'Moje ime');
$smarty->display('predlozak.tpl');
?>
```

Datoteka predloška koja će se koristiti ima naziv 'predlozak.tpl'. Uobičajeno je za Smarty datoteke predložaka koristiti ekstenziju tpl. Datoteka predloška mora biti smještena u mapi templates/.

Nakon izvođenja ovog programa sustav predložaka Smarty će pokušati otvoriti datoteku templates/predlozak.tpl i u njoj će u svim varijablama 'ime' upisati zadanu vrijednost 'Moje ime' te će zatim ispisati predložak. Rezultat ispisa će zamijeniti <?php ... ?> dio u samoj HTML datoteci.

Može se vidjeti da se pri korištenju predložaka Smarty ne trebaju koristiti naredbe echo i print kao što je to potrebno inače bez njih.

Datoteka 'predlozak.tpl' može izgledati ovako:

```
<html>
<head>
     <title>Prvi Smarty predložak</title>
</head>
<body>
Dobrodošli na moje stranice. Vi ste moj najdraži posjetitelj. Moje ime
je {$ime}.
</body>
```

Datoteka predlozak.tpl sadrži cijeli HTML dokument. Ona se od uobičajenog HTML dokumenta razlikuje samo po dijelu {$ime}. Predlošci Smarty koriste oznaku vitičaste zagrade {} kako bi označili dio koji će se mijenjati. Svi ostali dijelovi ispisuju se nepromijenjeni.

Unutar oznaka {} upisana je vrijednost $ime. Ova oznaka označava da će se ovdje upisati vrijednost varijable 'ime' koja je definirana u jeziku PHP.

Uloga programa pisanih u jeziku PHP pri upotrebi predložaka Smarty

Programi u programskom jeziku PHP se koriste isključivo kako bi definirali vrijednosti varijabli koje će se dalje ispisivati putem predložaka. Sam programski kôd u pravilu ne sadrži nikakve ispise već se svi podaci ispisuju isključivo putem predložaka. U nekim slučajevima može se činiti jednostavnijim podatke ispisati izravno u programskom jeziku PHP, ali time se gubi prednost upotrebe predložaka.

Sami programi u jeziku PHP nemaju nikakvih PHP oznaka i kad bi im se izravno pristupilo putem poslužitelja weba bez uključenih predložaka Smarty oni ne bi ništa ispisali iako bi se izvršili. Cijela njihova uloga je odrediti varijable pomoću metode assign() i zatim putem metode display() pozvati prikaz stranice. Osim ovog dodatka programi pisani u jeziku PHP ne trebaju imati nikakva saznanja o sustavu predložaka koji se koristi.

Velika prednost je što se na ovaj način sustav predložaka može i promijeniti ukoliko se za to pokaže opravdana potreba bez potrebe za izmjenom samih programa. Također, nije potrebno promijeniti niti jednu liniju programa u slučaju da je podatke potrebno drugačije vizualizirati.

Podaci se u predloške trebaju prenositi bez ikakvih grafičkih oznaka, a sva vizualizacija, kao što su definiranje oznaka slova, tablica i sličnog treba se raditi isključivo u samim predlošcima.

Osnove predložaka Smarty

Korištenje komentara

Osim mogućnosti jednostavnog ispisivanja vrijednosti pojedine varijable Smarty raspolaže i s brojnim funkcijama i dodacima koji olakšavaju rad s ispisom. Osnovna od mogućnosti je upotreba Smarty komentara. Ovi komentari se, za razliku od HTML komentara neće ispisivati u HTML datoteci već će se nalaziti isključivo u predlošku. Komentar počinje oznakom spojene otvorene vitičaste zagrade i zvjezdice: {*, a završava sa spojenom oznakom zvjezdice i zatvorene vitičaste zagrade *}. Na primjer:

```
Tekst HTML stranice.
{* Ovo je komentar u HTML stranici. *}
Nastavak teksta
{* Komentar u
     Više
       Linija *}
Kraj teksta
```

Rezultat ovog predloška u pregledniku će biti:

```
Tekst HTML stranice.
Nastavak teksta
Kraj teksta
```

Konfiguracijska datoteka

Predlošci Smarty mogu učitavati konfiguracijske datoteke koje mogu sadržavati varijable definirane prema odjeljcima i globalne varijable.

Primjer konfiguracijske datoteke:

```
#Ovo je konfiguracijska datoteka

# globalne varijable
nazivStranice = "Naslovna stranica"
bojaTeksta = #000000

# varijable vezane uz odjeljak
[Odjeljak1]
nazivStranice = "Informacije o glavnom odjeljku"

# varijable vezane uz odjeljak
[Odjeljak2]
nazivStranice = "Informacije o drugom odjeljku"
```

Broj konfiguracijskih datoteka nije ograničen, a one se učitavaju izravno iz samih predložaka upotrebom naredbe: {config_load file="datoteka.conf"} pri čemu je „datoteka.conf“ naziv konfiguracijske datoteke koja se želi učitati.

Varijable

Smarty raspolaže sa vrlo složenim sustavom ispisivanja vrijednosti varijabli. Na raspolaganju su ovi oblici upisivanja varijabli:

```
{* Ispisuje vrijednost varijable koja nije polje ili objekt *}
{$tekst}

{* Ispisuje vrijednost 5 elementa u polju $tekst *}
{$tekst[4]}

{* Odgovara obliku $tekst['polje'] u jeziku PHP *}
{$tekst.polje}

{* Odgovara obliku $tekst[$vrijednost] u jeziku PHP *}
{$tekst.$vrijednost}

{* Ispisuje svojstvo objekta 'vrijednost' u objektu $objekt *}
{$objekt->vrijednost}

{* Ispisuje vrijednost izvršavanja metode 'metoda' u objektu $objekt *}
{$objekt->metoda()}

{* Ispisuje vrijednost varijable tekst definirane u Smarty
 * konfiguracijskoj datoteci
 *}
{#tekst#}
```

Sve ove varijable, osim zadnje, definirane su u programu pisanom u programskom jeziku PHP. Posljednji oblik {#tekst#} koristi vrijednost varijable definirane u konfiguracijskoj datoteci.

Osim ovih postoje i varijable koje sam predložak Smarty definira, a putem kojih se mogu dohvatiti neke sistemske vrijednosti. Pristup sistemskim vrijednostima moguć je putem ugrađene varijable $smarty koja je raspoloživa u samom sustavu predložaka. Primjer dohvata ovih varijabli:

```
{* ispisuje vrijednost varijable $_GET['korisnik'] *}
{$smarty.get.korisnik}

{* ispisuje vrijednost varijable $_POST['korisnik'] *}
{$smarty.post.korisnik}

{* ispisuje vrijednost varijable $_COOKIE['korisnik'] *}
{$smarty.cookies.korisnik}

{* ispisuje vrijednost varijable $_SERVER['SERVER_NAME'] *}
{$smarty.server.SERVER_NAME}

{* ispisuje vrijednost varijable $_env['PATH'] *}
{$smarty.env.PATH}

{* ispisuje vrijednost varijable $_SESSION['korisnik'] *}
{$smarty.session.korisnik}

{* ispisuje broj sekundi od 1.1.1970 *}
{$smarty.now}

{* ispisuje inačicu predložaka Smarty *}
{$smarty.version}
```

Funkcije

Sustav predložaka Smarty ima mogućnost korištenja funkcija u svom izvršavanju. Inicijalno su u same predloške uključene neke gotove funkcije, ali se vrlo brzo i jednostavno mogu dodavati i nove, vlastite. Sve funkcije koriste se na jednak način:

```
{funkcija parametar1=vrijednost1 parametar2=vrijednost2}
```

U ovom primjeru pozvat će se funkcija pod nazivom „funkcija“ s dva parametra. To su parametri „parametar1“ i „parametar2“. Vrijednost „parametra1“ je „vrijednost1“, a „parametra2“ je „vrijednost2“. Broj i naziv parametara funkcije nije ograničen.

Primjer jedne od funkcija je i funkcija config_load koja ima parametar file. Na primjer:

```
{config_load file='postavke.conf'}
```

U ovu postoje i druge ugrađene funkcije. Jedna od najčešćih je funkcija if. Na primjer:

```
{if $prijavljen}
    Dobrodošli, <font color="{#fontColor#}">{$ime}!</font>
{else}
    Nažalost ne znam tko ste.
{/if}
```

Funkcije se mogu koristiti i za jednostavne matematičke operacije. Na primjer:

```
Vi ste posjetitelj: {$brojac+1}
Ukupna površina naše stranice je {$sirina*$visina}px.
```

Promjena vrijednosti varijabli

Sustav predložaka Smarty raspolaže sa mogućnosti promjene vrijednosti varijabli. Ova mogućnost se koristi kada je potrebno napraviti manje izmjene, a ne želi se mijenjati sam PHP izvorni kôd. Promjena vrijednosti se odnosi prvenstveno na nizove znakova iako je moguće definirati ovu mogućnost i za brojeve.

Promjena vrijednosti se koristi tako što se nakon naziva varijable upiše okomita crta i naziv vrste promjene varijable. Na primjer: {$tekst|capitalize}

U predlošcima Smarty na raspolaganju su promjene vrijednosti varijabli iz primjera. U svim primjerima vrijednost varijable $tekst je „Ovo je primjer teksta“.

Ispisivanje teksta velikim slovima:

```
{* OVO JE PRIMJER TEKSTA *}
{$tekst|capitalize}

{* Ovo Je Primjer Teksta *}
{$tekst|capitalize:true}
```

Dodaje niz znakova na kraj:

```
{* Ovo je primjer teksta koji je nadodan *}
{$tekst|cat:'koji je nadodan'}
```

Ispisuje broj znakova u nizu:

```
{* 21 - broji i razmake*}
{$tekst|count_characters}

{* 18 - ne broji razmake *}
{$tekst|count_characters:true}
```

Broji broj odlomaka. Prekidom odlomka se smatra oznaka \n:

```
{* 1 *}
{$tekst|count_paragraphs}

{* $primjer = „Tekst\nU odlomcima“ *}
{* 2 *}
{$primjer|count_paragraphs}
```

Broji rečenice u nizu znakova. Krajem rečenice se smatraju interpunkcijski znakovi kraja rečenice:

```
{* 1 *}
{$tekst|count_sentences}

{* $primjer = „Tekst. U - rečenicama.“ *}
{* 2 *}
{$primjer|count_sentences}
```

Broji riječi u rečenici:

```
{* 4 *}
{$tekst|count_words}
```

Oblikuje broj sekundi od 1.1.1970. u datum prema zadanom obliku. Postavke datuma odgovaraju onima u jeziku PHP:

```
{* Npr: Dec 1, 2012 *}
{$now|date_format}

{* Npr: Monday, December 1 2021*}
{$now|date_format:“ "%A, %B %e, %Y"}

{* Npr: 16:03:23 *}
{$now|date_format:“%H:%M:%S“}
```

Ako varijabla sadrži neku vrijednost ispisuje tu vrijednost inače ispisuje niz znakova koji je zadan kao parametar:

```
{* Ovo je primjer teksta *}
{$tekst|default:“Nije zadano“}

{* Nije zadano *}
{$prazna_varijabla|default:“Nije zadano“}
```

Ispisuje vrijednost varijable prikladan za URL:

```
{* Ovo je primjer teksta *}
{$tekst|escape}

{* $primjer = „Ovo je primjer 'teksta'“ *}
{* Ovo je primjer &#039;teksta&#039; *}
{$tekst|escape}

{* Ovo+je+primjer+&#039;teksta&#039; *}
{$tekst|escape:'url'}
```

Ispisuje odgovarajući broj razmaka ili drugih znakova kojima se uvlači tekst:

```
{*    Ovo je tekst *}
{$tekst|indent}

{*         Ovo je tekst *}
{$tekst|indent:10}

{* >>>Ovo je tekst *}
{$tekst|indent:3:'>'}
```

Ispisuje tekst pisan malim slovima:

```
{* ovo je tekst *}
{$tekst|lower}
```

Oznaku novog reda (\n) mijenja u
:

```
{* ovo je tekst *}
{* {$tekst|nl2br}

{* $primjer = „Tekst\nU odlomcima“ *}
{* Tekst<br />U odlomcima *}
{$primjer|nl2br}
```

Zamjenjuje nizove znakova koristeći regularne izraze:

```
{* Ov- j- t-kst *}
{$tekst|regex_replace:'/[oe]/':'-'}
```

Jednostavno zamjenjivanje niza znakova drugim nizom znakova:

```
{* Ovo je super *}
{$tekst|replace:'tekst':'super'}
```

Umeće razmak ili drugi znak između svaka dva znaka:

```
{* O v o   j e   t e k s t *}
{$tekst|spacify}

{* O_v_o_ _j_e_ _t_e_k_s_t *}
{$tekst|spacify:'_'}
```

Oblikuje ispis broja:

```
{* $primjer = 15.64232 *}
{* 15.64 *}
{$primjer|string_format:'%.2f'}

{* 16 *}
{$primjer|string_format:'%d'}
```

Briše sve dvostruke razmake, prijelome odlomaka sa jednim razmakom ili zadanim nizom znakova:

```
{* $primjer = 'Ovo     je    \n  tekst' *}
{* Ovo je tekst *}
{$primjer|strip}

{* Ovo_je_tekst *}
{$primjer|strip:'_'}
```

Briše sve HTML oznake iz niza znakova ili ih mijenja sa razmakom:

```
{* $primjer = 'Ovo je važan <b>tekst</b>' *}
{* Ovo je važan  tekst *}
{$primjer|strip_tags}

{* Ovo je važan tekst *}
{$primjer|strip_tags|false}
```

Ako je tekst duži od zadanog niza znakova skraćuje ga. Podrazumijevana vrijednost nakon koje skraćuje tekst je 80 znakova:

```
{* Ovo je primjer teksta *}
{$tekst|truncate}

{* Ovo je... *}
{$tekst|truncate:11}

{* Ovo je *}
{$tekst|truncate:11:''}

{* Ovo je___ *}
{$tekst|truncate:11:'___'}

{* Ovo je prim *}
{$tekst|truncate:11:'':true}

{* Ovo je p... *}
{$tekst|truncate:11:'...':true}

{* Ovo ..eksta *}
{$tekst|truncate:11:'':true:true}
```

Ispisuje niz znakova pisan velikim slovima:

```
{* OVO JE PRIMJER TEKSTA *}
{$tekst|upper}
```

Prelama dugačak tekst u nekoliko redaka. Podrazumijevana dužina je 80 znakova:

```
{* Ovo je primjer teksta *}
{$tekst|wordwrap}

{* Ovo je
   primjer
   teksta *}
{$tekst|wordwrap:6}

{* Ovo je#primjer#teksta* }
{$tekst|wordwrap:6:"#"}
```

Većina ovih izmjena varijabli ima i dodatne parametre pomoću kojih je moguće dodatno definirati njihove postavke.

Moguće je i kombinirati promjene varijabli. Na primjer:

```
{* OVO JE... *}
{$tekst|upper|truncate:11}
```

Ugrađene funkcije

Sustav predložaka Smarty raspolaže s nekoliko ugrađenih funkcija. Osim ovih moguće je definirati i vlastite funkcije. Funkcije se mogu koristiti za različite stvari. Od jednostavnog prolaza kroz polje čemu služi funkcija {foreach}, provjeri vrijednosti varijabli – određivanju toka izvršavanja pomoću funkcije {if} do uključivanja drugih datoteka funkcijom {include}.

Neke funkcije kao što su {php} i {include_php} omogućavaju i pisanje PHP kôda iako se to ne savjetuje budući da nije u skladu s filozofijom odvajanja poslovnog od prezentacijskog sloja čemu služi sustav predložaka.

Funkcije kao parametar mogu prihvaćati neki dio HTML dokumenta, a mogu i samo raditi s onim podacima koji su im parametri. Funkcije koje kao parametar mogu prihvaćati cijeli HTML dokument nazivaju se blokovskim funkcijama. Primjer takve funkcije je {literal} dok je primjer funkcije koja ne prihvaća HTML dokument kao parametar funkcija {include}.

Blokovske funkcije svoj blok započinju nazivom funkcije. Na primjer {literal} dok završavaju oznakom {/literal} odnosno, dodavanjem znaka / prije imena funkcije. Blok HTML dokumenta koji se nalazi unutar tih oznaka prenosi se kao parametar funkciji koja ga može modificirati ako je to potrebno.

U sklopu predložaka Smarty na raspolaganju su ugrađene funkcije iz primjera.

Blokovska funkcija koja rezultat izvođena bloka pohranjuje u varijablu.

```
{capture name=info}
     Danas je {$smarty.now|date}. Dobrodošli na ovaj poslužitelj.
     Vaša IP adresa je: {$smarty.server.REMOTE_ADDR}.
{/capture}

<div id='Info'>
  {$info|upper}
</div>
```

Omogućava učitavanje konfiguracijske datoteke. Osim jednostavnih postavki konfiguracijska datoteka se može koristiti i za pohranu jezičnih datoteka koje sadrže poruke na odgovarajućem jeziku:

```
{*    Sadržaj datoteke konfiguracija.conf:
      #Ovo je konfiguracijska datoteka

      # globalne varijable
      nazivStranice = "Naslovna stranica"
      bojaTeksta = #000000

      # varijable vezane uz odjeljak
      [Odjeljak1]
      nazivStranice = "Informacije o glavnom odjeljku"

      # varijable vezane uz odjeljak
      [Odjeljak2]
      nazivStranice = "Informacije o drugom odjeljku"
*}

{config_load file='konfiguracija.conf'}

{* Naslovna stranica *}
{#nazivStranice#}

{config_load file='konfiguracija.conf' section='Odjeljak1'}

{* Informacije o glavnom odjeljku *}
{#nazivStranice#}
```

Blokovska funkcija {foreach} omogućuje prolaz kroz elemente polja neovisno o tome da li je polje numerička ili imenovana.:

```
{* $polje = array(
     'Zagreb' => 10000,
     'Rijeka' => 51000,
     'Split' => 21000
   );
*}

{foreach from=$polje item=post_broj}
     {* Ispisuje:
          Poštanski broj: 10000
          Poštanski broj: 51000
          Poštanski broj: 21000 *}
     Poštanski broj: {$post_broj}
{foreachelse}
     {* Ispisuje se u slučaju $polje = array() *}
     Nema podatka o gradovima.
{/foreach}


{foreach from=$polje item=post_broj key=grad}
     {* Ispisuje:
          Poštanski broj grada Zagreb: 10000
          Poštanski broj grada Rijeka: 51000
          Poštanski broj grada Split: 21000 *}
     Poštanski broj grada {$grad}: {$post_broj}
{foreachelse}
     {* Ispisuje se u slučaju $polje = array() *}
     Nema podatka o gradovima.
{/foreach}
```

Određivanje toka programa pri čemu uvjeti za usporedbu odgovaraju onima u jeziku PHP pri čemu se koristi blokovska funkcija {if}:

```
{if $korisnik == 'Pero'}
     Dobrodošao Pero, kako si danas?
{elseif $korisnik == 'Hrvoje' || $korisnik == 'Petar'}
     Kako si mi ti?
{elseif $godine < 15}
     Djeluješ mi mlado.
{else}
     Ne znam te, ali to nije važno. Dobrodošao!
{/if}
```

Omogućava uključivanje drugih datoteka predložaka u ovu datoteku:

```
{include file='zaglavlje.tpl'}
     Ovo je moja stranica!
{include file='podnozje.tpl'}
```

Uključuje PHP datoteku koja se izvršava unutar predloška. Ova mogućnost postoji iz potrebe za sukladnosti sa prethodnim inačicama predložaka Smarty i ne savjetuje se njeno korištenje budući da nije sasvim u skladu s filozofijom odvajanjem poslovnog od prezentacijskog sloja:

```
{include_php file='/putanja/do/datoteke.php}
```

Ova funkcija ima sličnu funkcionalnost kao i funkcija {include}, ali se rezultat njenog izvršavanja neće pohranjivati u cache/ mapu. Vrijednost parametra 'name' je naziv funkcije koja će se pozvati iz PHP programa s prefiksom insert_:

```
{* Pozvat će funkciju insert_brojac iz PHP programa *}
{insert name='brojac'}
```

Funkcije {ldelim} i {rdelim} će ispisati Smarty oznaku za početak, odnosno kraj Smarty oznake. Podrazumijevana vrijednost je { za početak pisanja oznake, a } za kraj pisanja oznake. Podrška za ove funkcije je dodana budući da Smarty omogućava promjenu podrazumijevanih oznaka { i } u bilo koji niz znakova:

```
{* Ispisat će se: {,} *}
{ldelim},{rdelim}
```

Blokovska funkcija {literal} omogućava upotrebu Smarty oznaka za pisanje funkcija unutar bloka bez da ih Smarty pokušava analizirati kao Smarty oznake. Ova funkcija se najčešće koristi kod JavaScript funkcija gdje se oznake { i } koriste pri definiciji funkcije:

```
{literal}
   <script language="JavaScript" type="text/javascript">
     function umnozak(a,b){
       if(a < b) {
         return a*b*2;
       } else {
         return a*b;
       }
     }
   </script>
{/literal}
```

Blokovska funkcija PHP omogućava izravno uključivanje PHP naredbi u sam predložak. Ne savjetuje se upotreba ove funkcije, osim u iznimnim slučajevima, budući da nije u skladu s filozofijom odvajanja prezentacijskog od poslovnog sloja:

```
{* Ispisuje vrijednost globalnog brojača kojeg uvećava za 1 *}
{php}
   global $globalni_brojac;
   $globalni_brojac++;
   echo $globalni_brojac;
{/php}
```

Blokovska funkcija {section} omogućava prolaz kroz sve elemente numeričkih polja. Vrlo često se koristi prilikom dvodimenzionalnih polja:

```
{* $polje = array(1 => 'Zagreb', 2 => 'Rijeka', 3 => 'Split'); *}

{section loop=$polje name=i}
     {* Ispisuje:
          Grad: Zagreb
          Grad: Rijeka
          Grad: Split
      *}
     Grad: {$polje[i]}
{sectionelse}
     {* Ako je $polje = array() (prazno) *}
     Nema podatak u polju.
{/section}

{* Section funkcija se može koristiti i kao for petlja *}
{* Ispisuje: 15 17 19 21 23 25 27 29 *}
{section name=brojac start=15 loop=30 step=2}
  {$smarty.section.brojac.index}
{/section}
```

Blokovska funkcija {strip} briše sve suvišne razmake iz bloka:

```
{* Ovo je primjer nekog teksta sa HTML <b>oznakama</b> *}
{strip}
     Ovo je     primjer
         nekog                teksta
  sa HTML <b>oznakama</b>
{/strip}
```

Instalacija

WAMP

WAMP je besplatni program pod GPL licencom, koji olakšava instalaciju najnovijih inačica većine potrebnih programa za web programiranje ili aplikacije koje za svoj rad trebaju PHP i MySQL.

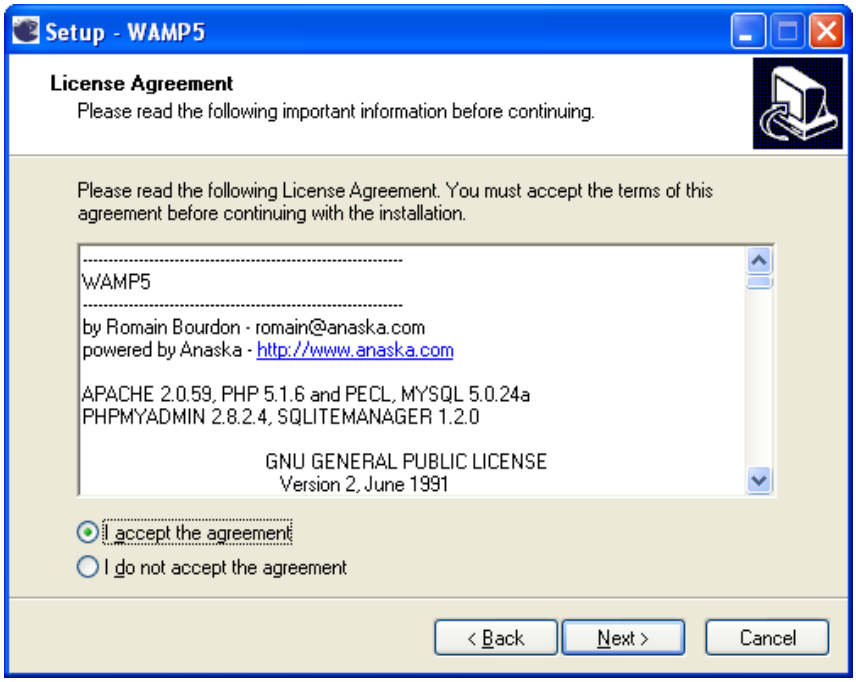
Instalacijom programa WAMP na računalu ćete moći koristiti sljedeće alate:

- **Apache** - Web poslužitelj (eng. server)
- **PHP5** - programski jezik
- **MySQL** - baza podataka
- **PHPmyadmin** – web aplikacija za jednostavno administriranje baze podataka
- **SQLitemanager** – jednostavna verzija baze podataka, za vrlo malo podataka

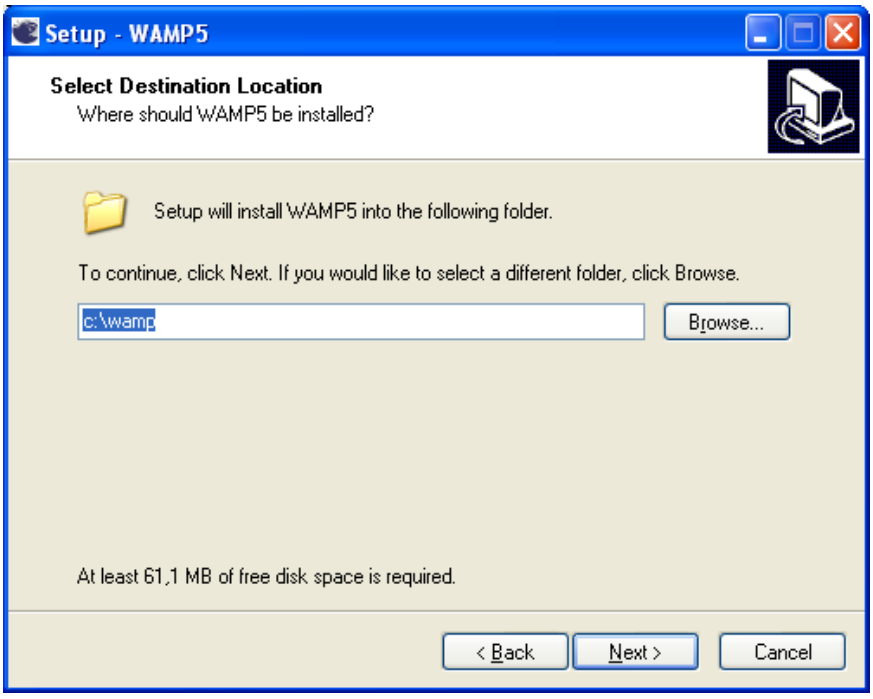
Koraci instalacije

Instalacija programa može se preuzeti s web adrese: http://www.wampserver.com/en/download.php. Nakon pokretanja instalacijske datoteke, sama instalacija se sastoji od nekoliko jednostavnih koraka:

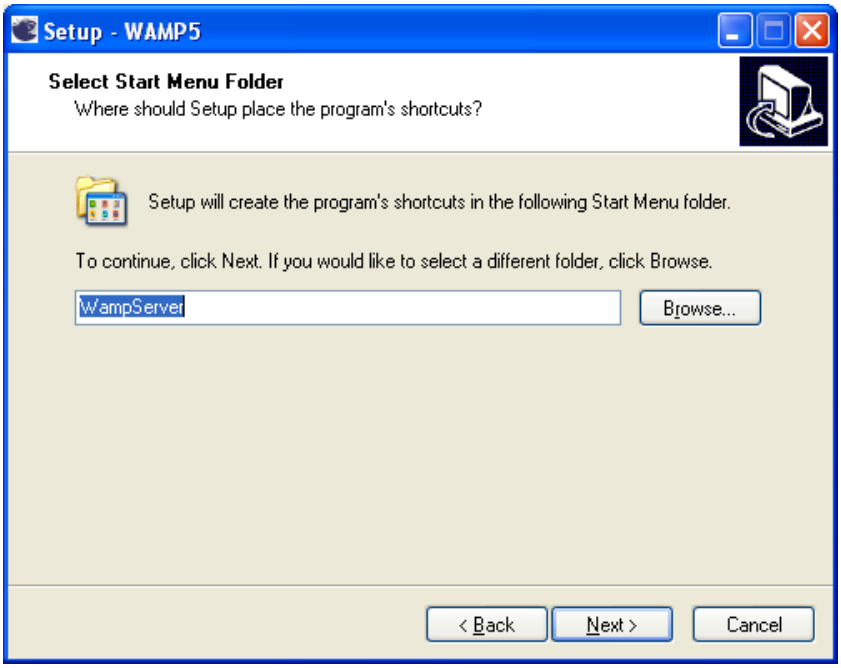
Pozdravni ekran s porukom o inačici i imenu programa koji se instalira. Za nastavak instalacije potrebno je odabrati opciju „Next >“



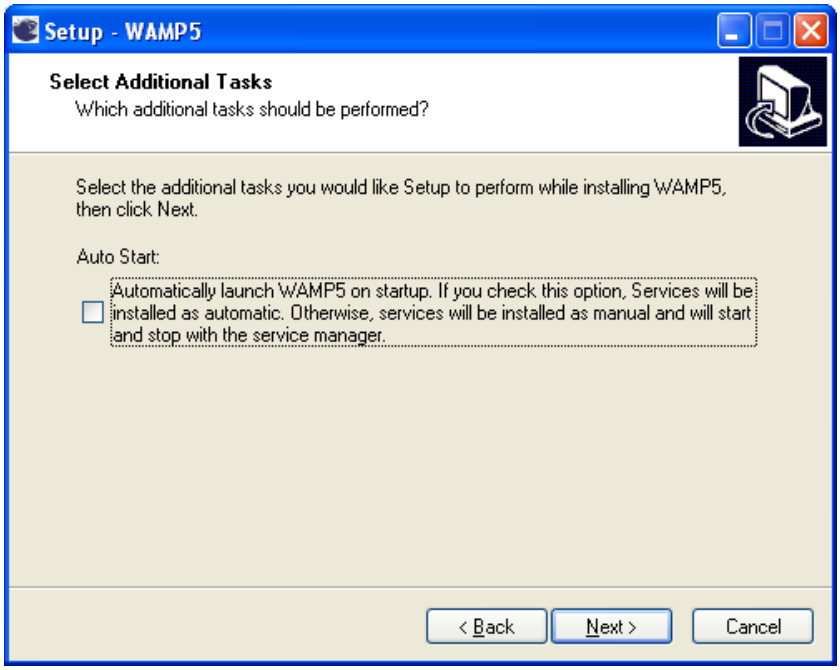
U ovom koraku ispisuje se licencija i uvjeti korištenja programa. Za nastavak instalacije potrebno je pročitati licenciju i odabrali mogućnost *"I accept the agreement"*, prelazak na sljedeći korak instalacije potvrđuje se odabirom mogućnosti "Next >".



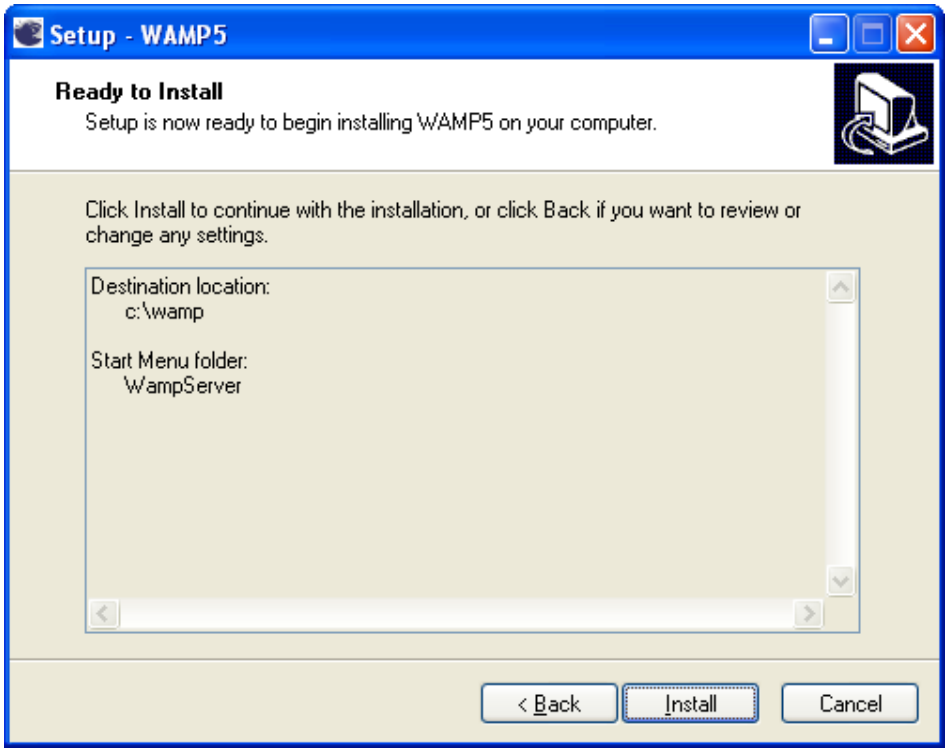
U ovom koraku instalacije nudi se mogućnost odabira mape u koju se želi sve instalirati. Za odabir nove mape potrebno je odabrati mogućnost „Browse" i odabrati novu mapu. Za nastavak instalacije potrebno je odabrati mogućnost "Next >".



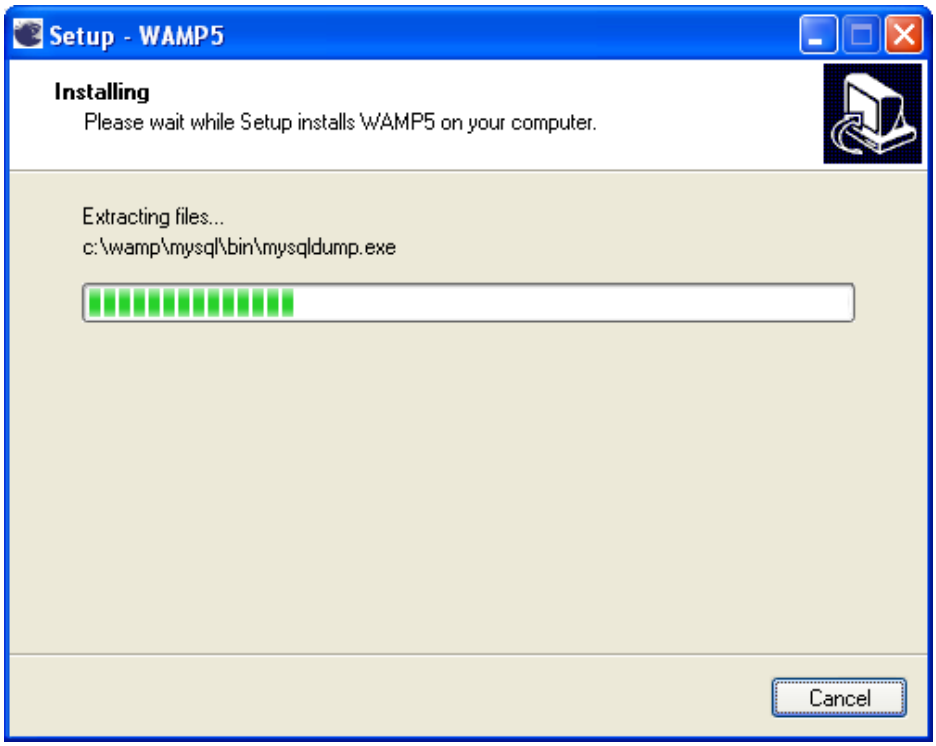
Prije prelaska na idući korak potrebno je odrediti naziv mape u koji se žele smjestiti prečice, a koja se nalazi u startnom izborniku.



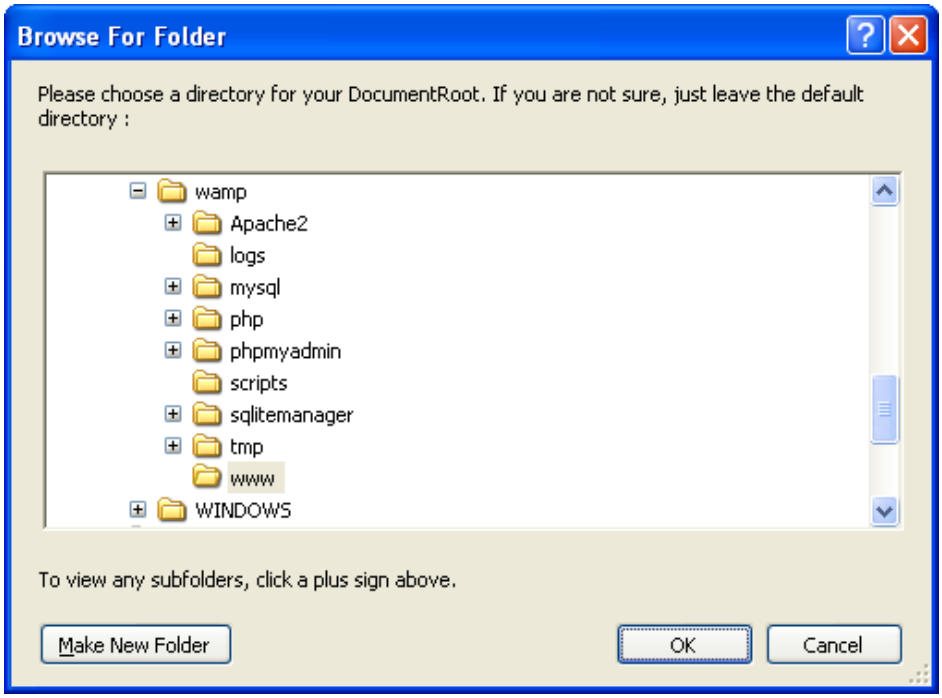
U ovom koraku potrebno je odlučiti da li je potrebno prilikom svakog podizanja sustava automatski pokrenuti servise za web (odabir ako je računalo namijenjeno kao serversko računalo za web aplikacije), ili ne (ako se sam isprobavaju programi). Ako želite automatsko pokretanje servisa , označite polje „Auto Start“, a za nastavak instalacije odaberite opciju „Next >“.



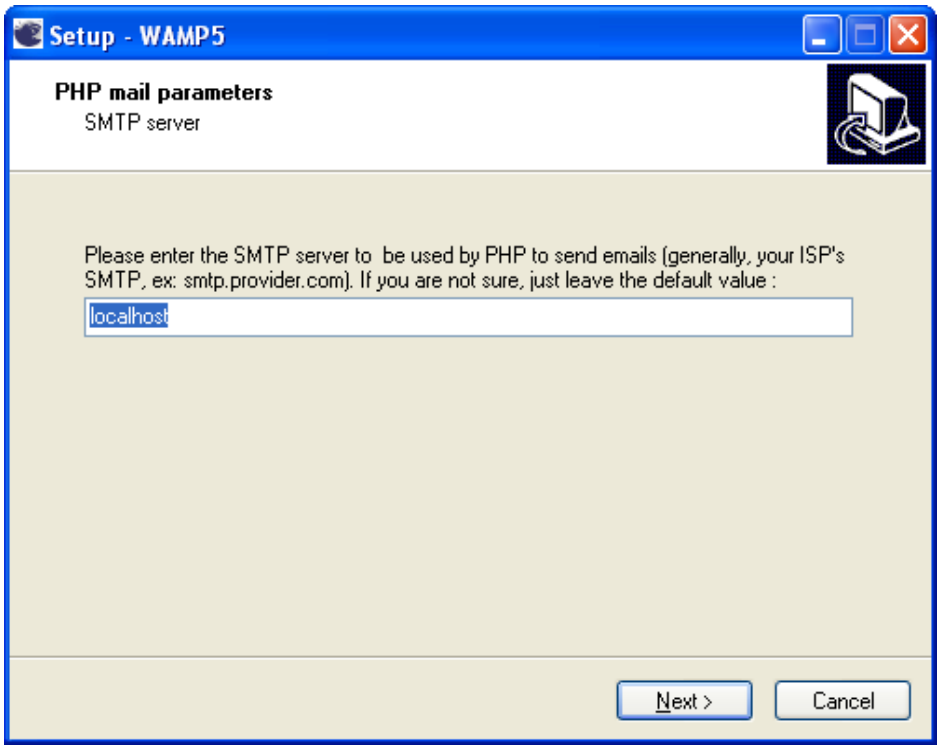
Program obavještava da je završio sa svim pripremama potrebnim za samu instalaciju. Odabirom mogućnosti "Install" pokrenut će se sama instalacija.



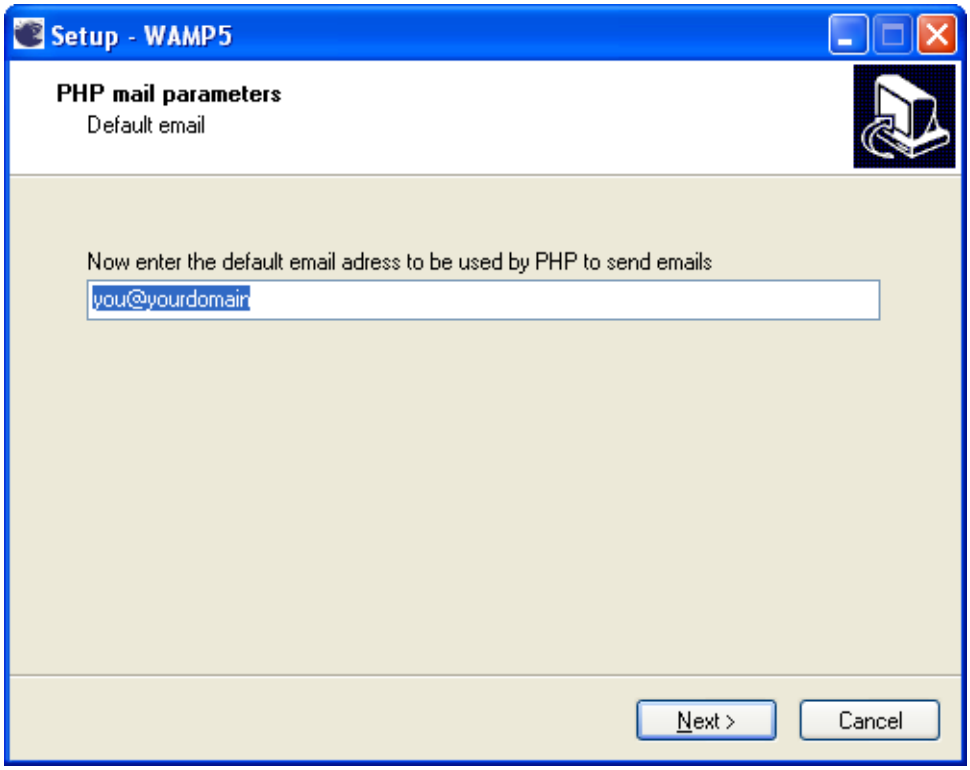
Sama instalacija traje vrlo kratko, a status instalacije moguće je pratiti kroz pomicanje statusne trake.



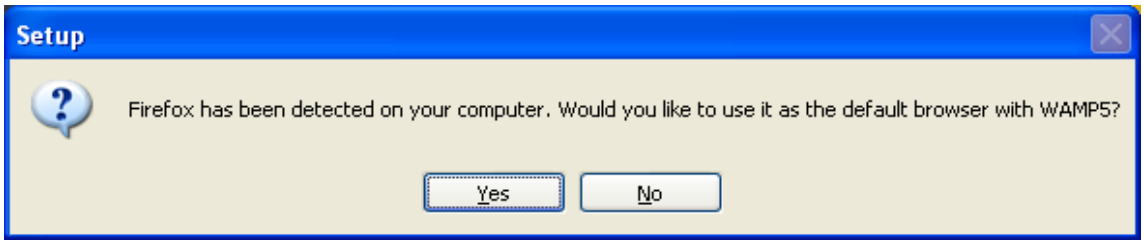
Nakon što je sama instalacija završila, potrebno je podesiti još nekoliko postavki. Na ovom ekranu potrebno je odabrati početni web direktorij ili ostaviti predloženi, te odabrati mogućnost „OK“ za nastavak.



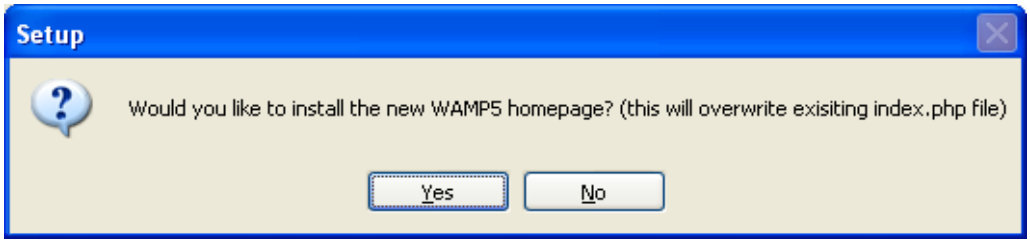
Na ovom ekranu potrebno je upisati naziv mail servera putem kojeg će se PHP slati email poruke ako se koristi odgovarajuća funkcija.



U ovom koraku potrebno je upisati email adresa pošiljatelja koju će PHP upisivati u svaki poslanu email poruku. Nakon što ste upisali e-mail adresu, odaberite opciju za nastavak instalacije „Next >“.



Ako je na sustavu već instaliran FireFox preglednik moguće ga je odabrati kao podrazumijevani preglednik s pristupom administracijskom sučelju. Za odabir preglednika Firefox potrebno je odabrati mogućnost „Yes“ za nastavak, a za odabir podrazumijevanog preglednika potrebno je odabrati mogućnost „No“.



U ovom koraku, moguće je odabrati da li da WAMP5 postavi vlastitu inicijalnu

datoteku u početni web direktorij ili ne. Ako je početni web direktorij prazan dobro je odabrati „Yes“, a ako sadrži već neke web stranice dobro je odabrati mogućnost „No“.

Nakon uspješne instalacije ispisuje se poruka o uspješnosti, a mogućnost ”Finish” će završiti instalaciju.

Literatura

[1] D. Lane, Hugh. E. Williams , **„Web Database Application with PHP and MySQL, 2nd Edition”**, *O'Reilly*, Svibanj 2004

[2] L. Gheorghe, H. Hayder, J. P. Maia, **„Smarty PHP Template programming and Application”**, *Packt Publishing*, Travanj 2006.