PHP
PHP i MySQL
PHP i Forme

PHP

Uvod u PHP

Jeste li spremni za revoluciju u svom životu kao web developera? Jeste li spremni žrtvovati par sati svog vremena da upoznate PHP sa svim njegovim vrlinama i manama? Uvjeravam Vas da nećete požaliti svoju odluku i da će slijedećih par sati, koliko će biti potrebno da Vas uvedem u predivan svijet PHP programiranja, biti dobro ulaganje u vlastitu budućnost.

Dobro ulaganje? Zašto? Vrlo jednostavno, PHP je jedan od najpopularnijih i najmoćnijih skriptnih jezika trenutno na tržištu. Broj siteova koji koriste PHP raste iz dana u dan, a broj tvrtki koje žele primijeniti PHP na svojim siteovima je još veći. Pa zašto onda ne biste i vi bili jedan od rijetkih koji može stvoriti jedan ovakav site?

Maloprije sam rekao 'predivan svijet PHP programiranja' . Sigurno se pitate što je tako predivno u njemu. Predivno je to što pomoću njega možete stvoriti opširnu web aplikaciju sa velikim količinama podataka sa takvom lakoćom da ni sami nećete vjerovati da ste to učinili. Samo si pokušajte zamisliti koliko bi ste se namučili kada bi išli raditi site koji bi trebao prezentirati paletu proizvoda neke firme i da ga krenete izrađivati u čistom HTML-u. Uzmimo da dotična firma ima u svom asortimanu oko 1000 proizvoda. To znači da bi morali napraviti 1000 stranica (za svaki proizvod posebnu) te bi ste morali paziti da svaka od njih izgleda isto ostalima (da su tablice poravnate, da su naslovi iste veličine i boje…). Ovo bi bio vrlo mukotrpan i stresan posao, a recimo da želite omogućiti svojim posjetiteljima vrlo jednostavno pretraživanje asortimana proizvoda ovaj zadatak bi postao praktički neizvediv!

Što bi rekli kada bih vam ja rekao da ovakav site možete napraviti u duplo kraćem roku sa trostruko većom funkcionalnošću? Da sam lud? A ne, ja nisam lud i ovo je vrlo moguće. A što bi ste rekli kada bih vam rekao da to možete izvesti sa potpuno besplatnim alatom u vašem omiljenom tekstualnom editoru (npr. Notepad)? Sada već mislite da sam na nekakvom opojnom sredstvu i da sam izgubio doticaj sa realnosti. Opet vas uvjeravam, sve što sam rekao stoji i nalazi se pred vama. Samo se morate malo potruditi i zgrabiti priliku koja leži ispred vas i iskoristiti ju najbolje što možete.

Što je PHP?

Php je open-source server-side **skriptni programski jezik** za dinamičko generiranje HTML koda.

Drugim riječima, PHP je skriptni programski jezik pomoću kojeg možete kreirati HTML stranicu na serveru prije nego što je ona poslana klijentu popunjenu dinamičkim sadržajem. Govorimo o radu sa templateima. Ovim načinom generiranja sadržaja klijent ne može vidjeti kod (skriptu) koji je generirao sadržaj koji gleda, već ima pristup čistom HTML kodu.

Open-source u gornjoj definiciji znači da svatko tko želi može skinuti izvorne PHP kodove pisane u C-u i, ukoliko ih razumije, može ih mijenjati po svojoj volji te dodavati nove funkcije PHP-u. Štoviše, svi su pozvani da sudjeluju u razvoju novih verzija PHP-a. Izvorne kodove i instalacijske datoteke možete skinuti sa službenog PHP sitea.

Server-side programiranje i usporedba PHP-a sa ostalim server side tehnologijama

PHP je jedna od najnaprednijih i najkorištenijih server-side skriptnih tehnologija danas u upotrebi. On je svojom sintaksom sličan mnogim drugim sličnim jezicima, čak i ima istoznačne (iste po sintaksi i funkcionalnosti) funkcije kao i neki drugi jezici kao što su C ili Perl. To znači da jednu radnju možete izvesti korištenjem više različitih funkcija.

Recimo ova dva primjera rezultiraju istim prikazom:

Primjer 1

```
<?

echo 'Pozdrav svima';

?>

i



<?

printr( 'Pozdrav svima');

?>
```

Rezultat predviđate da će izgledati:

```
Pozdrav svima
```

Još jedna važna stvar svima onima koji planiraju jednog dana biti napredni korisnici PHP-a je ta da je PHP bogat funkcijama za manipuliranje mnogo različitih tipova sadržaja. Od manipuliranja grafikom (png, jpg, flash…) do loadanja .NET modula i rada sa XML-om

Ono što PHP stavlja još više ispred ostalih web skriptnih tehnologija je njegova podrška za baratanje širokom paletom baza podataka. Podržava sve popularnije baze podatak kao MySQL, PostgreSQL, dBase, Oracle, ODBC…

Isto tako njegova neovisnost o operacijskom sustavu i pristupačne cijene (besplatan je) ga čini među prvim izborom velikih i malih kompanija za izradu vlastitih mrežnih sustava

Mislim da je ovo dovoljno da se svatko sa malo petlje i vremena upusti prekrasni svijet PHP programiranja.

PHP u usporedbi sa drugim server side skriptnim jezicima

PHP vs ASP

Prva razlika je ta što ASP sam po sebi nije jezik već skupina povezanih objekata kojima možemo pristupiti pomoću VB ili Java Scripta. Druga stvar koja stavlaj ASP u drugi plan je ta što je ASP podržan na Win32 sistemima sa IIS-om, a morali bi izdvojiti izdašnu sumu da bi ga pokrenuli na drugim platformama ili serverima. Veliki problem kod ASP-a je to što je programer ograničen samim jezikom koji je podijeljen na 'komponente' i ukoliko želimo šire mogućnosti morali bismo nadograđivati ASP dodatnim komponentama, što u Microsoft žargonu znači dodatni troškovi!

PHP vs Perl

Ova dva jezika imaju dosta sličnosti, ali više razlika. Glavna razlika je ta što je PHP namijenjen prvenstveno za web skriptanje a Perl ima mnogo širu upotrebu. Samim time ima i složeniju sintaksu od PHP-a što ga čini teže za naučiti i manje 'shvatljivim' od PHP-a. Unatoč široj uporabi Perla PHP koristi mnoge 'dobre' značajke Perla poput konstruktora (više o njima poslije) i nekih sintaktičkih osobina .

PHP vs Cold Fusion

Osnovna razlika između ova dva jezika je ta što je PHP stvoren sa misli na pravog programera sa iskustvom u nekom C stil jeziku, dok su stvoritelji Cold Fusiona imali u prvom planu neprogramere i njima su prilagodili sintaksu. Isto tako, PHP je pouzdaniji i otvoreniji inovacijama i programerskim trikovima od CF-a

Koja je razlika između server-side i client-side skriptnih jezika?

Server-side

Server side skripte se izvršavaju na serveru (poslužitelju) kada poslužitelj primi zahtjev za PHP dokumentom. Nakon primitka zahtjeva sa PHP dokumentom poslužitelj izvršava PHP kod i na osnovu njega generira HTML kod i šalje ga klijentu. To znači da stranica koja se prikazuje u pretraživaču klijenta ne postoji u tom obliku nigdje na serveru odakle ju je klijent primio. Ovo može stvoriti male poteškoće pri pozicioniranju vaših stranica na nekim tražilicama, ali postoje članci na netu koji objašnjavaju i ovu problematiku. Mi se njome nećemo baviti ovom prilikom.

Client-side

Glavni i najpoznatiji predstavnik ove skupine jezika je JavaScript. Kod pisan u JavaScriptu je obično umetnut u HTML stranicu i izvršava se tek u klijentovom pretraživaču. Ovakav kod vidljiv je svima, osim ako nemate malo iskustva i spremite svoje kodove u nekakav include file te time sakrijete svoje kodove koje ste razvijali u sitne noćne sate.

Možda bi bilo najbolje malo detaljnije razmotriti razlike između njih na jednom banalnom primjeru. Recimo da želimo napisati skriptu koja prikazuje točno vrijeme negdje na stranici. Iako ovo zvuči kao vrlo banalan i lagan primjer, ako ga malo bolje razmotrite vidjet ćete da je ovo gotovo nemoguće izvesti. Zašto? Prvo pitanje koje si trebate postaviti je 'Koje je vrijeme točno vrijeme?'. Ono na vašem serveru ili ono na klijentovom računalu? Što god odlučili, ove dvije skripte će rijetko kada prikazati isti rezultat.

Primjer 2

```
<script language="JavaScript">

var v = new Date()

document.write(v.getHours())

document.write(":")

document.write(v.getMinutes())

document.write(":")

document.write(v.getSeconds())

</script>
```

i

Primjer 3

```
<?

$str_vrijeme= date("H:i:s");

echo $vrijeme;

<?
```

Instalacija

Prije nego što krenete sa čitanjem ovih uputa, možda bi trebali posjetiti službeni PHP site (**www.php.net**) i tamo skinuti potrebne fileove.

Trenutna stabilna verzija je 4.1, a postoje prijedlozi za verziju 4.2, ali nisu predviđeni za upotrebu, već samo za testiranje.

Tipovi instalacije:

Najviša na popisu je instalacija pomoću source kodova i ukoliko nemate iskustva sa C-om i kompajliranjem nemojte se upuštati u to.

Druga, puno prihvatljivija i preporučena je instalacija putem InstallShielda. Ovo je čarobnjak koji će vas provesti kroz cijeli proces instalacije i ukoliko pratite upute nećete naići na nikakve probleme. Tijekom čarobnjaka možete izabrati standardnu ili naprednu instalaciju. U standardnoj vas čarobnjak neće previše gnjaviti sa pitanjima i instalirat će PHP na IIS ili PWS server bez problema. U naprednoj instalaciji možete sami izabrati neke osobine PHp-a. Čarobnjak će automatski konfigurirati vaš serverski software i namjestit će php.ini file koji je potreban za funkcioniranje PHP-a u vaš system root direktorij. Oba tipa su provjerena i rezultat je garantiran tako da ovdje neću ulaziti u detalje.

Treći tip instalacije je putem zip arhive. Ovo je najmoćniji tip instalacije. Njime možete instalirati neke dodatne ekstenzije kao što su GD library (za manipulaciju grafičkih dokumenata) i slične module. Popis ekstenzija koje se nalaze u arhivi naći ćete na službenom siteu PHP-a na dijelu koji objašnjava instalaciju na Win operacijskom sistemu. Što se tiče same instalacije slijedite ovih par koraka:

1. Stvorite PHP folder (c:\php)
2. U njega odzipajte arhivu. Sada se u PHP folderu nalaze svi fileovi potrebni za rad PHP-a uključujući php.exe te dva različita php.ini filea.
3. Odaberite jedan od php.ini fileova (jednostavnosti radi odaberite php.ini-recommended, skinite mu iz imena recommended i prebacite ga u vaš system root folder (npr c:\win)
4. Tip instalacije koji sam odabrao za ovaj priručnik je instalacija CGI aplikacije. Naime ovaj tip je najstabilniji (SAPI i ISAPI instalacija, iako je malo funkcionalnija, može stvarati probleme pri radu tako da ću to ostaviti vama naprednim korisnicima da se sami poigrate. Početnicima i nije od neke velike važnosti imati PHP instaliran kao SAPI modul – više informacija o SAPI i ISAPI modulima nađite na **www.php.net**)
5. Da bi osigurali rad PHP-a morate se pobrinuti da se svi dll-ovi mogu naći u vašem sistemu. Zato ih možete ostaviti u istom direktoriju kao i php.exe ili ih prebaciti u <sistem root>\system32 ili <sistem root>\system . Dll koji je potreban za rad PHP-a kao CGI aplikacije je php4ts.dll i on se već nalazi u istom folderu kao i php.exe
6. Ukoliko želite koristiti dodatne ekstenzije nađite u php.ini fileu [extension_dir] liniju i u njoj navedite lokaciju extensions direktorija (C:\PHP\extensions) i maknite ' ; ' ispred onih ekstenzija koje želite koristiti. Proučite na PHP siteu koje ekstenzije dolaze uz distibuciju jer ukoliko odkomentirate neku koja se ne nalazi u distribuciji vaš server neće raditi.

7. Zadnji korak (pri radu sa IIS 4.0 ili više) je povezivanje .php datoteka sa php.exe fileom. Ukoliko preskoćite ovaj korak IIS neće znati što treba učiniti sa .php datotekama. Otvorite IIS, označite Default web site i kliknite Properties. U odjeljku Home directory odaberite Configuration. U prozoru koji se otvorio odaberite Add. U odgovarajuće polje unesite path php.exe izvršnog filea a pod Exstension upišite .php. Ponovite ovu proceduru i za .php3 ekstenziju.

8. Ukoliko naiđete na nekakve probleme pri radu otvorite install.txt file gdje je ova procedura detaljnije obješnjena.

Sintaksa PHP-a

Već ste u prijašnjem primjeru mogli vidjeti neke bitne stvari. Npr. da se sav PHP kod nalazi između <? i ?> kvačica. Mali dodatak ovom pravilu bi bio korištenje <?php … ?> kvačica radi razlikovanja između PHP i XML koda (naime i XML koristi iste ove kvačice). Druga stvar koja je očita iz tog primjera je da varijable prije svog imena imaju znak $.

Prelaženje iz PHP u HTML mode

Vrlo bitna karakteristika PHP-a i bilo kojeg drugog jezika je razdvajanje server side koda od statičkog HTML-a. Tako u PHP-u korištenjem <? I ?> govorimo serveru da se između njih nalazi PHP kod i da je potrebno prvo njega izvršiti i tek nakon toga poslati HTML output tog koda skupa sa ostatkom statičkog HTML koda klijentu. Ovo je osnova server-client mrežne komunikacije putem TCP/IP protokola i ne bih ulazio u detalje, ali ono što je korisno znati je kako ovo koristiti u vlastitu korist i olakšati sebi posao u razvijanju PHP aplikacija.

Unutar koda se možemo u bilo kojem trenutku prebaciti iz HTML moda u PHP mode. Čak i unutar if, for i ostalih kontrolnih struktura. Ovo nam omogućuje rad sa templateima u kojima se na isti način prikazuju različiti podatci iste strukture. Zvuči nejasno? Da ilustriram jednim vrlo jednostavnim primjerom, a detaljniji opis dolazi kasnije u vodiču.

Primjer 4

```
<html>

<head>

<title>Untitled</title>

</head>

<body>

<?

$ime="Marko";

$prezime="Markic";

?>

<table width="300" border="1" cellspacing="0" cellpadding="0">
```

```
<tr>

<td>

<?

echo $ime;

?>

</td>

<td><?=$ime?></td>

</tr>

</table>

</body>

</html>
```

Rezultat gornjeg koda bi izgledao

Marko	Markić

Uočite da sam za ispis sadržaja varijable koristio dva različita načina koji su rezultirali istim krajnjim ispisom. Naime unutar <? i ?> se nalazi php kod koji će se izvršiti ukoliko se u njemu ne nalaze neke sintaktičke pogreške. Druga metoda ispisa je izgledala ovako :

```
<?=$prezime?>
```

Ovu metodu koristimo kada želimo ispisati neku varijablu ili string. Ova linija je identična

```
<? echo $prezime ?>
```

samo što smo umjesto echo naredbe koristili znak = koji PHP-u govori da ispiše ono što se nalazi između <? i ?> kvačica. Više o ovim metodama u poglavlju 'Tehnike PHP programiranja'

Varijable

Kao što sam već spomenuo varijable prije svog imena obavezno moraju sadržavati znak **$**. Ovo je čisto način govorenja PHP prevoditelju da se radi o varijabli a ne o tekstu. Ukoliko ga izostavite

aplikacija će javiti grešku (u najboljem slučaju), a preći će preko nje (u najgorem slučaju) i umjesto sadržaja varijable će ispisati samo njeno ime.

Još jedna vrlo bitna stvar kod varijabli u PHP-u je da su imena varijabli case-sensitive. Siguran sa da ste svi upoznati sa značenjem ovog pojma ali da vam ilustriram ovo pravilo možda bi bio dobar jedan mali primjer:

```
$mojeime ? $MojeIme
```

Isto tako, u imenima varijabli ne smijete koristiti razmake niti bilo kakve znakove osim [i] koji se koriste u nizovima i kod nekih metoda rada sa stringovima, ali o tome malo kasnije, te znaka '_'. Svi ostali znakovi su zabranjeni u imenima varijabli. Isto tako, ime varijable ne smije početi sa brojem, ali ga može sadržavati na bilo kojoj drugoj poziciji u imenu.

U PHP-u se ne morate brinuti o brisanju varijabli radi štednje memorijskog prostora jer se sve varijable automatski brišu iz memorije kada se trenutna skripta koja ih je stvorila završi. Ukoliko stvarno želite obrisati neku varijablu iz bilo kojeg razloga to možete učiniti pomoću unset($varijabla); naredbe

Validna imena varijabli

```
$str_ime | $varijabla2 | $niz[1] | $string[0]
```

NE validna imena varijabli

```
$2varijabla | $var*ijabla | $_var | $var(1)
```

Tipovi podataka

U PHP-u ne postoje fiksni tipovi podataka. Naime, ne morate definirati tip varijable prije njenog korištenja i varijablu možete deklarirati bilo kada unutar skripte i pridruživati joj različite tipove podataka tokom izvođenja skripte (iako ovo nije uobičajena niti previše pametna praksa). Isto tako možete mijenjati tip podataka neke varijable jednog te istog sadržaja, ali o tome par redaka kasnije.

Tipovi podataka koje podržava PHP su:

- Cijeli brojevi (integer)
- Realni brojevi (floating-point numbers)
- Tekstualni podatci (String)

- Logičke varijable
- Nizovi
- Objekti

U slijedećim primjerima koristit ću neke funkcije koje će vam možda biti nepoznate i neće vam odmah biti jasno čemu one služe. Njih zanemarite i pokušajte shvatiti primjere što bolje možete. Sve funkcije će biti detaljnije objašnjene malo kasnije, kada bude bilo priče o kontrolnim strukturama i sličnim stvarima.

Cijeli brojevi

U ovaj tip varijable možemo pohraniti pozitivne i negativne brojeve u rasponu od

-2147483648 do 2147483647 tj. 32 bita podataka. Možemo ih zapisati u decimalnom, oktalnom ili heksadecimalnom zapisu.

Par primjera:

```
$int_var=123; //pozitivan decimalni broj

$int_var=-123; //negativni decimalni broj

$int_var=0123; //oktalni broj

$int_var=0#123; //heksadecimalni broj
```

Realni brojevi

Postoje dva načina spremanja realnih brojeva

```
$dbl_var=0.123;

// ili

$dbl_var=1.123e8;
```

Budite pažljivi kada koristite realne brojeve. Naime njihova točnost nije garantirana (ima veze sa pretvaranje ovog broja u njegov binarni ekvivalent. Recimo 0.33333 nikada neće biti točno prebačen u binarni ekvivalent). Stoga, nemojte ih uspoređivati za jednakost i vjerovat im do posljednje decimale.

Tekstualni podaci

Sadržaj string tipa varijable se nalazi između navodnika. Možete koristiti duple i jednostruke navodnike. Postoje razlike u ispisu sadržaja ovisno o tipu navodnika koje koristite.

Korištenjem **duplih navodnika** možete koristiti 'spcial characters'. To su posebni znakovi koji govore PHP-u da izvrši određene radnje pri ispisu sadržaja varijable. Ako ste ikada radili u C-u ili Perlu već ste upoznati za ovim znakovima. To su znakovi koji slijede iza znaka backslash (\ ). On se ujedno koristi za preskakanje određenog znaka unutar stringa.

Lista escape znakova:

Znak	**Značenje**
\n	Novi red(LF ili 0x0A u ASCIIu)
\t	Tab razmak (HT ili 0x09 u ASCIIu)
\\	Backslash
\$	Dolar znak
\"	Dupli navodnik

Ovi znakovi neće imati utjecaja na izgled same stranice u prozoru browsera, već će njihov utjecaj biti vidljiv tek pri pregledu sourcea dokumenta. Ovime možete sasvim sakriti činjenicu da je stranica stvorena putem PHP-a i pomoću njih je lakše pronaći grešku u generiranoj stranici. U protivnom bi se sav sadržaj ispisao u jedan red bez razmaka. Uviđate da bi bilo vrlo teško u tom neredu naći bilo što, a kamoli grešku u ispisu, ako ju tražite u source viewu.

Ukoliko želite da se neki tekst prebaci u novi red pri gledanju stranice u prozoru browsera morat ćete se poslužiti
 i sličnim tagovima. Znači, ako se želite koristiti PHP-om morate se jako dobro znati služiti HTML-om.

Još jedan bitna razlika između duplih i jednostrukih navodnika je ta da će se pri korištenju duplih navodnika sadržaj varijable ispisati a pri korištenju jednostrukih navodnika ispisat će se ime varijable skupa sa znakom $. Mali primjer bi ovo dobro ilustrirao.

Primjer 5.

```
<?

$str_ime="Kreso";

echo ("Moje ime je $str_ime");



?>
```

Ispisuje:

```
moje ime je Kreso
```

Dok će,

Primjer 6.

```
<?

$str_ime="Kreso";

echo ('Moje ime je $str_ime');

?>
```

ispisati:

```
moje ime je $str_ime
```

U PHP-u je također moguće spajanje više stringova u jedan ispis. To radimo pomoću '.' . Evo primjera:

Primjer 7.

```
<?

$str_var1='Dijete';

$str_var2='ide'; //nema razmaka prije ili poslije riječi

echo $str_var1 . ' ' . $str_var2 . ' u školu';
```

```
// rezultira sa

// Dijete ide u školu

?>
```

Uočite da su u gornjem primjeru korišteni jednostruki navodnici, ali varijable se ne nalaze u njima tako da će se njihov sadržaj ispisati. Također su izostavljene zagrade, što je dozvoljeno.

Kao što sam već spomenuo, pri radu sa stringovima možemo u njihovom imenu koristiti znakove [i]. Njih koristimo kada želimo izdvojiti određeni znak iz stringa, tj. String zamislimo kao jednodimenzionalni numerički niz indeksiran na taj način da se na svakom broju, počevši od 0, nalazi jedan znak stringa. Raspon indexa je od 0 do n-1; gdje je n broj znakova niza.

```
<?

$str_tekst='Dijete ide u školu';

$str_znak=$str_tekst[0];

echo $str_znak; //ispisuje 'D'

echo $str_tekst[3]; //ispisuje 'e'

echo $str_tekst[strlen($str_tekst)-1] //ispisuje zadnji znak 'u'

?>
```

Kao što vidite u njih možete smjestiti bilo koji izraz koji će na kraju rezultirati cijelim brojem (integerom).
U ovom primjeru je korištena naredba
strlen($neki_string)

Koja vraća duljinu (broj znakova) nekog stringa.

Logičke varijable

Logički tip podataka ima dvije moguće vrijednosti : true i false

Deklaracija logičke varijable:

```
$logicka=true;
```

```
$logicka=false;
```

Ovaj tip podataka je također rezultat logičkih izraza (npr. iz if uvjeta) te nekih PHP funkcija.

Nizovi

PHP podržava više vrsta nizova. Tekstualne (associative) i cjelobrojne (vectors / indexed) indeksima. Mogu biti jednodimenzionalni ili multidimenzionalni.

Primjer cjelobrojnog jednodimenzionalnog niza: (Primjer 9)

```
<?

$arr_boje=array('plavo','žuto','zeleno');

echo $arr_boje[0]; // ispisat će 'plavo'

echo $arr_boje[2]; // ispisat će 'zeleno'

$arr_boje[3]='crveno'; // dodaje novi element u niz

$arr_boje[2]='ljubičasto'; // mijenja staru vrijednost na indexu 2 – zeleno prelazi
u ljubičasto

$arr_boje[7]='roza' // indexi ne moraju slijediti kronološki redosljed

// želite li ispisati sve elemente niza možete se služiti ovom metodom



foreach ($arr_boje as $int_kljuc => $str_vrijednost){

echo $int_kljuc . " => " . $str_vrijednost . "<br>\n";

}

?>
```

Rezultat :

```
plavo
zeleno
```

```
_________________
0=> plavo
1=> žuto
2=> ljubičasto
3=> crveno
7=> roza
```

Pirmjer 10: Stvaranje cjelobrojnog niza

```
<?

// ako želite petljom stvoriti niz od n elemenata gdje će svakom elementu biti
pridodan

// faktorijel njegovog indeksa to učinite ovako



$n=10; // niz će imati 10 elemenata

$int_faktorijel=1; // inicijalizacija faktorijela

for ($i=1;$i<=$n;$i++){

$int_faktorijel*=$i;



// ovdje smo mogli komotno koristiti i

//$int_faktorijel=$int_faktorijel * $i;



$arr_niz[$i]=$int_faktorijel; // elementu pridružujemo njegov faktorijel



// mogli smo koristiti i

// $arr_niz[]=$int_faktorijel;

// ali onda indeks ne mi odgovarao faktorijelu jer bi indeksi počeli sa 0 a završili
sa 9

}

foreach ($arr_niz as $int_kljuc => $int_vrijednost){

echo $int_kljuc . " => " . $int_vrijednost . "<br>\n";

}



?>
```

Rezultat :

```
1=> 1
2=> 2
3=> 6
4=> 24
5=> 120
6=> 720
7=> 5040
8=> 40321
9=> 362880
10=> 3628800
```

Primjer 11: Primjer tekstualnog (associative) niza

```
<?

// recimo da želite reproducirati sadržaj svog hladnjaka u niz

$arr_hladnjak=array(

"jaja"=>12,

"paprika"=>6,

"maslac"=>0,

"mlijeko"=>0.5,

"salama"=>"0.2 kg - narezano",

"sir"=>"0.4 kg - u komadu"

);

echo $arr_hladnjak["salama"]."<br>\n"; // ispisali ste koliko salame imate u
hladnjaku



// ili multidimezionalni



$arr_boje=array(

"tople"=>array("žuta","crvena"),

"hladne"=>array("plava","zelena")
```

```
);

// ako želite ispisati npr žuta

echo $arr_boje["tople"][0]."<br>\n";

?>
```

Rezultat :

```
0.2 kg -narezano
žuta
```

Možemo i kombinirati ova dva tipa niza

```
<?

$arr_kontakti = array(

"kreso"=>array("visina"=>182,"tezina"=>70,0=>"01/9876-543", 1=>"091/3432-876"),

"mirta"=>array("visina"=>164,"tezina"=>63,0=>"01/3256-937", 1=>"098/435-556")

);



echo $arr_kontakti["kreso"][0]."<br>\n";

echo $arr_kontakti["mirta"]["tezina"]."<br>\n";

?>
```

Funkcije

Funkcije u PHP-u definiramo slijedećom sintaksom bilo gdje u skripti.

```
<?

function funk($argument1,$argument2…$argumentN){

// … blok naredbi funkcije

}
```

```
?>
```

Primjer 14: Funkcije sa povratnim rezultatom:

```
<? function zbroji($broj1,$broj2){ $zbroj=$broj1+$broj2;
return $zbroj;
} // pridruživanje varijabli rezultata funkcije
$suma=zbroji(2,2);

// direktan ispis rezultata funkcije
$br1=12;
$br2=22;
echo zbroji($br1,$br2);
?>
```

U gornjem primjeru smo umjesto varijabli $br1 i $br2 koristiti i varijable imena $broj1 i $broj2. Ovo je moguće zato što funkciji ne dajemo same varijable već njihove vrijednosti koje se spremaju u lokalne (funkcijeske) varijable koje postoje samo za vrijeme izvođenja funkcije. Ukoliko želite funkcijom izmijeniti sadržaj neke varijable morate ju dati funkciji pomoću reference (*pass by reference*)

Primjer 15 : Davanje funkciji varijablu pomoću reference

```
<?

function povecaj(&$broj,$vrijednost){

$broj+=$vrijednost;

}



$neki_broj=12;

povecaj($neki_broj,10);

echo $neki_broj;

?>
```

Uočite da nije obavezno koristiti isto ime varijable u glavnom programu i funkciji, iako je moguće. Rezultat će u oba slučaja biti isti.

Primjer 16 : Funkcije sa default vrijednostima

```
<?
```

```
function frizider($stanje="pun"){

echo "Frizider je $stanje";

}



// ispis default vrijednosti

frizider();

echo "<hr>";



// ispis dane vrijednosti

frizider("prazan");

?>
```

Rezultat :

```
Frižider je pun
_________________

Frižider je prazan
```

Pir radu sa funkcijama koje imaju default vrijednost bitno je da ta vrijednost mora biti konstanta (string ili broj). Isto tako ukoliko funkcija ima više argumenata od kojih je jedan ili više imaju default vrijednosti oni moraju biti sa krajnje desne strane liste argumenata iza argumenata bez default vrijednosti:

Ovaj slučaj ne radi:

```
<?

function automobili($bolji="Audi",$gori){

echo "$bolji je bolji od $gori";

}

automobili("Peglice");

?>
```

Pravilno korištenje :

```
<?
function automobili($gori,$bolji="Audi"){

echo "$bolji je bolji od $gori";

}

automobili("Peglice");

?>
```

Objekti

Da, istina je. I PHP 4 podržava objektno programiranje. Iako ne onako kako ga podržava C++ ili slični jezici, ali svoje funkcije možete grupirati u klase te stvarati instance tog objekta kroz cijelu aplikaciju. Za sada se neću puno zadržavati na njima jer detaljnije objašnjenje slijedi u dijelu koji se bavi tehnikama PHP-a.

Sve u svemu, da bi ste koristili objekte prvo morate stvoriti klasu (class) sa nekim funkcijama i varijablama u njoj te ju pozivati unutar koda. Varijable unutar klase zamišljamo kao proporties tog objekta, a njene funkcije kao metode istog objekta.

Primjer 12 : Jednostavan objekt

```
<?

class objekt{

var $testna;

function ispisi(){

echo $this->testna;

}

}



// kreiranje instance objeka

$obj=new objekt;



// namještanje objektne varijable $testna

$obj->testna="Tekst koji je spreman za ispis";
```

```
// pozivanje funkcije koja za ispis

$obj->ispisi();

?>
```

Rezultat :

```
Tekst koji je spreman za ispis
```

U gornjem primjeru uz osnovne tehnike kreiranje objekta i njegove instance u skripti možete primijetiti još jednu stvar.

```
echo $this->testna;
```

Korištenjem $this-> prije imena varijable pristupamo toj varijabli i možemo ju koristiti za ispis ili izmjenu podataka. Na isti način pristupamo i funkcijama.

Ukoliko želite pri instanciranju objekta automatski izvršiti neke radnje poput spajanja na bazu podataka ili bilo koju sličnu operaciju koja je potrebna za daljnji rad objekta koristite **konstruktore**.

Konstruktori su funkcije objekta (klase) koji se izvršavaju zajedno sa stvaranjem instance objekta. Konstruktori imaju isto ime kao i sam objekt.

Primjer 13 : Objekt sa konstruktorom (dodatak primjeru 12)

```
<?

class objekt{

var $testna;

// konstruktor

function objekt(){

$this->testna="Ovo je tekst koji smo stvorili konstruktorom<br>";

$this->ispisi();

}

function ispisi(){

echo $this->testna;

}
```

```
}



// pozivanje objekta u kodu

$obj=new objekt;



// namještanje objektne varijable $testna

$obj->testna="Tekst koji je spreman za ispis";



// pozivanje funkcije koja za ispis

$obj->ispisi();

?>
```

Rezultat :

```
Ovo je tekst koji smo stvorili konstruktorom
Tekst koji je spreman za ispis
```

Za objekte i varijable i funkcije u njemu vrijede sva pravila kao i za normalne varijable i funkcije!

Imenovanje varijabli

Možda ste primijetili da sam u većini primjerima koristio sličan, ako ne isti, način imenovanja varijabli. Na prvom mjestu imena se nalazi opis tipa varijable od tri znaka. Nakon njega odvojeno sa '_' slijedi ime varijable koje bi trebalo pobliže opisati ime sadržaj same varijable. Ime se obično sastoji od jedne ili dvije riječi koje pobliže opisuju sadržaj varijable. Riječi možete odvajati sa znakom '_' ili svako početno slovo riječi možete napisati velikim slovom.

Zašto se pridržavati ovih pravila?

Pridržavanjem ovih pravili činite uslugu sebi jednako koliko i ostalima koji će jednog dana pokušati pročitati i razumjeti vaše kodove.

Pokušajte si zamisliti ovaj primjer :

Nalazite se između dva posla i odlučili ste razviti forum za vlastiti site. Krenete ga razvijati i u usred posla primite narudžbu za novi site. Normalno, trenutno zaboravljate na forum i bacate se na posao. Recimo da završite sa poslom za mjesec dana i sjetite se da ste prije posla izrađivali forum za vlastiti site. Otvorite svoj omiljeni editor, otvorite mjesto gdje ste stali kada ste ga zadnji put pogledali i… pred vama se sada nalazi hrpa slova i znakova koji vam više nemaju nekakvo značenje (ako ste koristili imena poput $post, $ime, $mail – čije ime, kakav post (tek stvoreni, već postojeći…)). Sada morate cijelu aplikaciju nanovo 'skužiti' i tek onda nastaviti sa radom.

Vrlo jednostavnom upotrebom standarda pri imenovanju varijabli i čestim komentiranjem koda ovih problema ne bi bilo. A zamislite da je netko drugi otvorio takve kodove. Trebalo bi mu par dana da shvati što koji red izvršava i što se nalazi u kojoj varijabli u određenom trenutku.

Stoga, pomognite sebi i drugima i pišite kodove sa komentarima i standardiziranim imenima varijabli. Vrijedno je truda.

Prebacivanje tipova varijabli

Sadržaj bilo koje varijable podložan je izmjeni svog tipa. Znači da nekakav broj može vrlo lako postati string i obratno.

Tipove možete mijenjati implicitno i eksplicitno (u slijedećim promjerima se neću pridržavati gore navedenih pravila imenovanja varijabli)

Primjer 17 – potrebno ga napisati

```
<?

// implicitno

$var=1; // varijabla je tipa integer

$var+=1.5 // varijabla je tipa double (realni broj)



$var = 5 + "10 Little Piggies"; // varijabla je integer (15)

$var = 5 + "10 Small Pigs"; // varijabla je integer (15)



// eksplicitno

$int_neki_broj=10;
```

```
$str_neki_broj= (string) $int_neki_broj // prebacili smo broj u string (ASCII
znakove)



?>
```

Eksplicitna izmjena tipa varijable se vrši tako da u neku novu varijablu pridružite neku već postojeću varijablu i ispred nje navedete u zagradama tip u koji želimo prebaciti varijablu koju pridružujete. Isto tako možete u varijablu pridružiti nju samu sa eksplicitnom izmjenom tipa podatka

```
<?

$int_var=10;

$int_var=(double) $int_var;

echo gettype($int_var); // gettype($int_var) vraća tip dane varijable

?>
```

Moguće konverzije su:

- (int), (integer) – prebaci u integer
- (real), (double), (float) – prebaci u double (realni broj)
- (string) – prebaci u string
- (array) – prebaci u niz
- (object) – prebaci u objekt

Operatori

Aritmetički operatori:

Primjer	Ime	Rezultat
$a + $b	Zbrajanje	Zbroj od $a i $b.
$a - $b	Oduzimanje	Razlika od $a i $b.
$a * $b	Množenje	Produkt od $a i $b.
$a / $b	Dijeljenje	Kvocijent od $a i $b.
$a % $b	Modul	Ostatak dijeljenja od $a i $b.

Mala napomena : Ukoliko dijelite dva broja koja su oba cjelobrojna i kvocijent će biti cjelobrojan. Ukoliko je jedna od varijabli realnog tipa i kvocijent će biti realnog tipa.

Operatori pridruživanja:

Primjer 18 – potrebno ga je napisati

```
<?

$int_var=5;

$int_var+=5; // sada je vrijednost varijable 10 – isto ako da smo napisali
$int_var=$int_var+5



$int_var*=5; // vrijednost varijable je 50

$int_var/=10 // vrijednost je 5



// kod stringova imamo

$str_tekst='Moje ime je ';

$str_tekst.='Kreso'; // sada je sadržaj varijable Moje ime je Kreso



//možete se igrati malo kompleksnijim izrazima poput

$a = ($b = 4) + 5; // rezultat je 9

?>
```

Logički operatori:

$a and $b	I	True ako su oboje $a i $b true.
$a or $b	Ili	True ako je $a true ili ako je $b true.
$a xor $b	Xor	ako je $a true ili ako je $b true, ali ne i ako su oba true.
! $a	Ne	True ako je $a false i obrnuto.
$a && $b	I	True ako su oboje $a i $b true.
$a ||	Ili	True ako je $a true ili ako je $b true.

$b

U gornjoj tablici $a ili $b mogu biti bilo koji izrazi koji vraćaju true ili false kao ishod svoje operacije. Tako da će slijedeći primjer biti ne samo dozvoljen, već i prijeko potreban.

```
<?

If ( ( ($int_var%2)==0) and ($int_var>10)){

// kod koji se izvršava samo ako je broj paran i veći od 10

} else {

// kod koji se izvršava ako je broj neparan ili ako je manji ili jednak 10 ili oboje

}

?>
```

Operatori uspoređivanja

Primjer	Ime	Rezultat
$a == $b	Jednako	True ako je $a jednako $b.
$a === $b	Identično	True ako je $a jednako $b, i ako su istog tipa.
$a != $b	Nije jedanko	True ako $a nije jednako $b.
$a !== $b	Nije identično	True ako $a nije jednako $b, i ako nisu istog tipa.
$a < $b	Manje	True ako je $a izričito manje od $b.
$a > $b	Veće	True ako je $izričito veće od $b.
$a <= $b	Manje jednako	True ako je $a manje ili jednako $b.
$a >= $b	Veće jednako	True ako je $a veće ili jednako $b.

Budite pažljivi pri traženju jednakosti dvije varijable da ne upišete $a=$b umjesto $a ==$b . Naime, u prvom slučaju izraz će vratiti true ako uspješno pridruži sadržaj varijable $b varijabli $a, a u drugom slučaju će vratiti true ako su jednake.

Operatori uvećavanja i smanjivanja

Primjer	Ime	Efekt
++$a	Preduvećavanje	Uveća $a za jedan, i onda vrati $a.
$a++	Naknadno uvećanje	Vrati $a, i onda ga uveća za jedan.
--$a	Predsmanjenje	Umanji $a za jedan, i onda vrati $a.
$a--	Naknadno smanjenje	Vrati $a, i onda ga umanji za jedan.

Kontrolne strukture

Pomoću kontrolnih struktura određujemo tok skripti, odlučujemo i računamo. One su zadužene za logiku aplikacija.

```
If.. else
```

If.. else je najčešće korištena kontrolna struktura. Njoj dajemo logički izraz koji se provjerava i ovisno o njegovom ishodu koji može biti true ili false izvršava se blok naredbi.

```
<?

If ( uvjet ) {

// naredbe koje se izvršavaju ukoliko je uvjet == true

} else {

// naredbe koje se izvršavaju ukoliko je uvjet == false

}

?>
```

Vitičaste zagrade ({}) označavaju blok naredbi. Njih možete izostaviti ukoliko grana ima samo jednu naredbu. Npr

Primjer 19 – potrebno ga je napisati

```
<?

$str_ime='Matija';

if ($str_ime=='Matija')

echo 'Bok matija';

else

die ('Ti nisi Matija. Ajde bok'); // die() je funkcija koja prekida izvršavanje
skripte. Ako joj u

// argument date neki tekst ili broj ispisat će ga. Također prima

// neku funkciju kao argument

?>
```

Umjesto else ključne riječi može se koristiti i elseif ključna riječ. Ona se izvršava ako je uvjet u if-u rezultirao false. Ona također ispituje logički izraz.

```
<?

If ( uvjet ) {

// naredbe koje se izvršavaju ukoliko je uvjet == true

} elseif ( uvjet2 ) {

// naredbe koje se izvršavaju ukoliko je uvjet == false i uvjet2==true

} elseif ( uvjet3 ) {

// naredbe koje se izvršavaju ukoliko je uvjet == false i uvjet2 == false i uvjet3
== true

} else {

// naredbe koje se izvršavaju ukoliko su svi uvjeti == false

}

?>
```

Ukoliko pokušavate riješiti situaciju sa mnogo mogućih ishoda nije praktično koristiti

if … elseif tip grananja. U tom slučaju koristite se switch strukturom.

Switch uzima za argument nekakav izraz i onda gleda da li je on jednak jednom od zadanih slučaja. Ukoliko nije jednak niti jednom od njih izvršava default akciju ili ne izvršava ništa.

Switch je idealan alat za izradu višenamjenskih stranica. Praktičnu primjenu ćete vidjeti u pokaznoj aplikaciji koju ćemo zajedno izraditi unutar ovog priručnika

```
<?



switch ( uvjet ){

case < slučaj1 >:

// naredbe koje se izvršavaju ukoliko je uvjet jednak slučaju 1

break;

case < slučaj2 >:

// naredbe koje se izvršavaju ukoliko je uvjet jedank slučaju 2

break;

case < slučaj3 >:

// naredbe koje se izvršavaju ukoliko je uvjet jedank slučaju 3

break;

default:

// naredbe koje se izvršavaju ukoliko uvjet nije jednak niti jednom slučaju

// njega se može izostaviti ukoliko se niti jedan naredba ne treba izvršiti u tom
slučaju

}

?>
```

Ključna riječ break označava završetak grane. Ukoliko ga izostavite između dvije grane naredbe obje grane će se izvršiti ukoliko je viša (ona koja slijedi prije) aktivirana. Naredbe će se izvršavati sve dok se ne pojavi break ili završetak switcha

Primjer 20 – potrebno ga je napisati

```
<?
```

```
$int_var=5;



switch ($int_var){

case 0:

echo 'Broj je nula';

break;

case 5:

echo 'Broj je pet';

case 6:

echo 'Broj je 6';

break;

default:

echo 'Broj nije poznat';

}

?>
```

Petlje

While petlja

While petlja izvršava svoj blok naredbi dokle god je izraz u uvjetu istinit (true). Uvjet se ispituje prije izvođenja bloka naredbi. Zbog toga je moguće da se blok ne izvrši niti jednom ukoliko je uvjet na početku false.

```
<?

while ( uvjet ) {

// naredbe koje se izvršavaju dok je uvjet true

}
```

```
?>
```

Ili konkretan primjer

Primjer 21 : - potrebno ga je napisati

```
<?

$int_var=10;

while ($int_var<=20){

echo '$int_var = '. ++$int_var. "<br>";

}



// rezultira sa

// $int_var = 11
// $int_var = 12
// $int_var = 13
// $int_var = 14
// $int_var = 15
// $int_var = 16
// $int_var = 17
// $int_var = 18
// $int_var = 19
// $int_var = 20
// $int_var = 21

?>
```

Do ... while

Za razliku od normalne while petlje, kod Do … while petlje uvjet se ispituje tek nakon izvršavanja bloka naredbi. Tako da je uvijek garantirano barem jedno izvršavanje bloka naredbi iako je uvjet odmah na početku false.

```
<?



do {
```

```
// naredbe koje se izvršavaju dok je uvjet true

} while ( uvjet )



?>
```

For petlja

For petlja koristi brojač petlje koji se prije svakog izvršavanja bloka naredbi petlje uveća ili smanji. For petlju koristite kada znate točan broj potrebnog ponavljanja bloka petlje. Brojač petlje može biti bilo koja već postojeća varijabla ili možete stvoriti novu varijablu za potrebe petlje. Ukoliko rabite drugi tip uobičajena imena takvih varijabli su $i, $j, $k i njih ćete sresti u gotovim svim aplikacijama diljem svijeta.

```
<?

for ($i=0;$i (operator uspuređivanja) (vrijednost sa kojom uspoređujete); (operator
uvećanja ili smanjenja){

// naredbe koje se izvrašavaju svki put dok je uvijet jednak true

}

?>
```

ili na konkretnom primjeru

Primjer 21: - potrebno ga je napisati

```
for ($i=10;$i>=0;$i--){

echo '$i = ' . $i . '<br>';

}
```

// što rezultira

```
// $i = 10
// $i = 9
// $i = 8
// $i = 7
// $i = 6
// $i = 5
// $i = 4
```

```
// $i = 3
// $i = 2
// $i = 1
// $i = 0
```

Foreach petlja

Foreach petlja se koristi za rad sa nizovima. Ona prolazi kroz svaki element danog niza i obavlja blok naredbi. Može spremiti ključ i vrijednost svakog elementa niza u posebne varijable u kojim se za svako ponavljanje petlje nalaze ključ i vrijednost elementa niza na kojem se trenutno nalazi nutarnji pokazivač. Nutarnji pokazivač se prije ulaska u petlju nalazi na 0 i svakim novim krugom u petlji povećava se za 1. Novim zvanjem foreach petlje nutarnji pokazivač se resetira. Petlja se vrti sve dok ne ostane bez elemenata niza.

```
<?



// općenito

foreach ($neki_niz as $vrijednost){

// naredbe koje se izvršavaju za svaki element niza

}



// ili



foreach ($neki niz as $kljuc => $vrijednost){

// naredbe koje se izvršavaju za svaki element niza

}



// evo jedan već korišten primjer



$arr_hladnjak=array(

"jaja"=>12,
```

```
"paprika"=>6,

"maslac"=>0,

"mlijeko"=>0.5,

"salama"=>"0.2 kg - narezano",

"sir"=>"0.4 kg - u komadu"

);

foreach ($arr_hladnjak as $kljuc => $vrijednost){

echo "$kljuc => $vrijednost <br>";

}



// rezultira

// jaja => 12
// paprika => 6
// maslac => 0
// mlijeko => 0.5
// salama => 0.2 kg - narezano
// sir => 0.4 kg - u komadu



?>
```

Komentari

PHP podržava više tipova komentara. U dosadašnjem tekstu sam se koristio inline komentarima (// komentar). Oni preskaču sav tekst koji se nalazi iza njih sve do početka novog reda.

Ukoliko želite komentirati više redova koristite se multiline komentarima

```
<?

/* Ovo je komentar

koje se proteže kroz čak

tri reda */
```

```
// ovo je komentar u jednom redu
$var=123; // a možemo ga koristiti i na kraju svakog reda


?>
```

Rad sa stringovima

Tehnike PHP programiranja

U ovom poglavlju bit će govora o osnovnim tehnikama rada u PHP-u. Kroz poglavlje upoznat ćete se sa glavnim funkcijama u PHP-u i steći osnovna znanja potrebna za izradu HTML aplikacija koje se baziraju na tekstualnoj reprezentaciji podataka i prikupljanju podataka iz formi. Također će biti govora o nekim malo naprednijim osobinama kao što su session managment i slanje e-maila.

Rad sa stringovima

Stringovi su najčešći tip podataka kojim se barata u PHP-u zbog prirode PHP-a kao alata za izradu HTML stranica na serveru. Iz ovog vrlo logičnog razloga bitno je znati baratati sa njima i manipulirati njihovim sadržajem. U PHP-u se nalaze već gotove funkcije koje automatiziraju većinu operacija koje su potrebne za manipuliranje i rad sa sadržajem string podataka.

Tehnika 1: Izbacivanje nepotrebnog 'bijelog prostora' iz stringa

Ponekada pri radu sa stringovima, a posebno pri obradi podataka iz formi, podatci u sebi sadrže nepotrebne razmake na početku ili kraju stringa. Pod razmacima se podrazumijevaju : razmak od jednog mjesta(space), tab razmak od više spaceova (escape znak \t), znak za novi red na kraju reda (escape znak \n)

Treba obratiti pozornost na to da se nepotrebni prazni prostor i rezultati escape znakova ne vide u HTML pogledu dokumenta već tek kada pogledate source dokumenta ili ukoliko se nalaze unutar <pre>…</pre> tagova za prikaz predformatiranog teksta.

Pogledajmo neke primjere:

Primjer 22: Izbacivanje nepotrebnog praznog prostora iz stringa

```
<?

$str_glavni=" \tOvo je string \nsa puno praznog prostora \n";
```

```
/* ispis stringa na ekran - prikazuje se normalno na ekran,

ali u source pogledu su razmaci vidljivi*/

echo $str_glavni;



// ispis stringa u <pre> tag. Sada su razmaci vidljivi

echo "<pre>$str_glavni</pre>\n";



// izbacivanje praznog prostora sa pocetka stringa

$str_bez_pocetka=ltrim($str_glavni);

echo "<hr>\n";

echo "<h4>String bez praznog prostora na pocetku stringa</h4>\n";

echo "<pre>$str_bez_pocetka</pre>\n";



// izbacivanje praznog prostora sa kraja stringa

$str_bez_kraja=rtrim($str_glavni);

echo "<hr>\n";

echo "<h4>String bez praznog prostora na kraju stringa</h4>\n";

echo "<pre>$str_bez_kraja</pre>\n";



// izbacivanjepraznog prostora iz cijelog stringa

$str_bez_praznina=trim($str_glavni);

echo "<hr>\n";

echo "<h4>String bez praznog prostora na kraju i pocetku stringa</h4>\n";

echo "<pre>$str_bez_praznina</pre>\n";

?>
```

Objašnjenje:

U ovom primjeru su korištene tri funkcije za izbacivanje praznina sa početka i kraja stringa.

Prva je ltrim(string)

ltrim izbacuje praznine i escape znakove sa lijeve strane (početka) stringa

Druga je rtrim(string)

rtrim izbacuje praznine i escape znakove sa desne strane (kraja) stringa

Treća je trim(string)

trim izbacuje praznine i escape znakove sa obje strane stringa

Ove funkcije mogu izbaciti praznine samo sa početka i kraja striga dok one unutar stringa ostaju na svom mjestu. Ukoliko njih želite izbaciti morat ćete koristiti druge metode poput explode() funkcije čije objašnjenje slijedi u Tehnici 3.

Tehnika 2: Traženje znaka ili stringa unutar stringa

PHP je opremljen funkcijama za pretraživanje string tipova podataka na više načina i metoda. Najjednostavnija metoda je korištenjem string funkcija, a sintaktički malo kompliciraniji način je korištenjem Regular expresiona. U ovoj tehnici je objašnjeno korištenje string funkcija.

Primjer 23 a): Pretraživanje stringa

```
<?

$string="Ovo je tekst u kojem se trazi neka rijec. Sastoji se od vise
recenica.\nTakoder ima i dva reda ilustracije radi";
$trazeni="je";

// ispis stringa kakav je

?>

<h3>Originalni string</h3>

<?

echo $string;

echo "<hr>\n";

?><h3>Traženje stringa ili znaka unutar zadanog stringa</h3><?

// trazenje nekog stringa ili znaka unutar stringa

$str_pretraga=strstr($string,$trazeni);

echo $str_pretraga;

?>
```

Objašnjenje:

U gornjem primjeru je za pretraživanje korištena funkcija strstr($str_zadani,$str_trazeni)

Njoj se kao prvi argument daje string u kojem želimo naći neku vrijednost i kao drugi joj dajemo vrijednost koju tražimo. Ukoliko postoji tražena vrijednost rezultat je ostatak stringa zajedno sa pronađenom vrijednosti na početku rezultata. Svi znakovi prije pojavljivanja su izbačeni.

Napomena : Ukoliko želimo ponovno koristiti ovaj 'skraćeni' string za ponovnu pretragu uvijek ćemo dobiti isti rezultat jer se nađeni string ili znak ne izbacuje iz rezultat već se stavlja na početak. Ukoliko želite opet pretražiti ovaj string

Primjer 23 b): Provjera postojanja stringa ili znaka unutar stringa

<hr>

<h3>Ispitivanje postojanja stringa ili znaka unutar zadanog stringa</h3>

```
<?

// provjeravanje da li postoji string ili znak unutra stringa

if($str_pretraga=strstr($string,"zzz")){

echo "Pronaden je string ili znak<br>".$str_pretraga;

} else {

echo "Trazeni string ili znak ne postoji u stringu";

}

?>
```

Objašnjenje:

```
if($str_pretraga=strstr($string,"zzz")){
```

Ovom linijom ispitujemo da li se tražena vrijednost nalazi u danom stringu. U isto vrijeme rezultat spremamo u varijablu za kasnije korištenje. Ovo je u potpunosti legalno ispitivanje jer funkcija strstr( ) vraća False ukoliko tražena vrijednost nije pronađena. Ovime se u varijablu sprema vrijednost False koju provjerava if struktura i izvršava se njen else blok koda. U slučaju da je vrijednost pronađena u varijablu se sprema rezultat funkcije koji if struktura gleda kao na true rezultat izraza u uvjetu.

Primjer 23 c): Pronalaženje pozicije prvog pojavljivanja znaka ili stringa u stringu i ispis ukupnog broja pojavljivanja.

```
<hr>

<h3>Ispisivanje pozicije pronadenog stringa ili znaka unutar zadanog stringa</h3>

<?

// ipsisivanje pozicije prvog nalazenja trazenog strina unutar zadanog stringa

$int_pozicija=strpos($string,$trazeni);

echo $int_pozicija;

?>

<hr>

<h3>Ukupan broj pojavljivanja traženog znaka ili stringa</h3>

<?

// racunanje broja pojavljivanja nekog stringa ili znaka unutar stringa

$suma=substr_count($string,$trazeni);

echo "Broj pojavljivanja :$suma";

?>
```

Objašnjenje:

```
// ipsisivanje pozicije prvog nalazenja trazenog strina unutar zadanog stringa

$int_pozicija=strpos($string,$trazeni);

echo $int_pozicija;
```

U ovom dijelu koda ispisujemo prvu poziciju pronađenog stringa u zadanom stringu. Za to se koristi funkcija strpos($str_zadani,$str_trazeni);

Funkcija vraća integer vrijednost koja predstavlja poziciju prvog znaka traženog stringa unutar zadanog stringa. Ukoliko trazeni string ne postoji unutar zadanog stringa funkcija vraća vrijednost False.

Napomena : Brojanje pozicije kreće od 0!!!

```
// racunanje broja pojavljivanja nekog stringa ili znaka unutar stringa

$suma=substr_count($string,$trazeni);

echo "Broj pojavljivanja :$suma";
```

Ovaj blok naredbi računa i ispisuje ukupan broj pojavljivanja traženog stringa unutar zadanog stringa. Za računanje se koristi funkcija substr_count($str_zadani,$str_trazeni)

Funkcija vraća inteher vrijednost koja reprezntira ukupan broj pojavljivanj traženog stringa unutar zadanog stringa. Funkcija vraća 0 ukoliko nema pojavljivanja. Ovo se interpretira kao False vrijednost unutar if strukture.

Primjer 23 d): Izvlačenje točno određenog dijela stringa

```
<hr>

<h3>Ispisivanje tocno odredenog dijela stringa</h3>

<?

// ispisvanje prvih 4 znaka stringa

$str_izvuceni=substr($string,0,4);

echo "Prva 4 znaka stringa : '$str_izvuceni'<br>";

// ispisivanje zadnja 4 znaka stringa

$str_izvuceni=substr($string,-4,4);

echo "Zadnja 4 znaka stringa : '$str_izvuceni'<br>";

// ispisivanje 10 znaka stringa

$str_izvuceni=substr($string,10,1);

echo "Deseti znak stringa : '$str_izvuceni'<br>";

// ispis 15 i svih ostalih znakova do kraja stringa

$str_izvuceni=substr($string,15);

echo "Svi znakovi od 15. znaka: '$str_izvuceni'<br>";

?>
```

Objašnjenje:

Za izvlačenje točno određenog dijela stringa koristi se funkcija
substr($str_zadani, $int_pocetna_pozicija, $int_broj_znakova)

Postoji više načina primjene ove funkcije, ovisno o komadu stringa kojeg želimo izvući.

```
// ispisvanje prvih 4 znaka stringa

$str_izvuceni=substr($string,0,4);

echo "Prva 4 znaka stringa : '$str_izvuceni'<br>";
```

Ovim načinom vadimo znakove od nultog znaka pa četiri slijedeća znaka. Ukoliko je potrebno izvući znakove sa kraja stringa funkciji dajemo kao početnu poziciji negativni broj.

```
// ispisivanje zadnja 4 znaka stringa

$str_izvuceni=substr($string,-4,4);

echo "Zadnja 4 znaka stringa : '$str_izvuceni'<br>";
```

Ovime govorimo PHP-u da počne izvlačiti znakove za zadanu vrijednost od kraja stringa, te da izvuče onoliko znakova od te pozicije koliko je zadano trećim argumentom.

Ukoliko treći argument ne postoji vraćaju se svi preostali znakovi od zadane pozicije do kraja stringa.

```
// ispis 15 i svih ostalih znakova do kraja stringa

$str_izvuceni=substr($string,15);

echo "Svi znakovi od 15. znaka: '$str_izvuceni'<br>";
```

Ukoliko je potrebo izvući samo jedan znak iz stringa funkciji kao treći argument dajemo 1 što će izvući samo znak sa zadane pozicije iz drugog argumenta.

```
// ispisivanje 10 znaka stringa

$str_izvuceni=substr($string,10,1);

echo "Deseti znak stringa : '$str_izvuceni'<br>";
```

U ovom slučaju smo mogli koristiti i drugu metodu koja je u nekim situacijama jednostavnija i praktičnija. Naime, string možemo zamisliti kao niz znakova i kako takvom možemo pristupati svakom znaku preko njegovog indeksa. Pozicije znakova tj. indeksi kreću od 0

```
// ispis 10 znaka stringa

echo $string[10];

Ova metoda se koristi kada želimo provjeravati svaki znak u stringu posebno. Da bi
izvršili ovakav način šetnja kroz string možemo koristiti dvije metode.

Prva je korištenjem for petlje:

$duljina=strlen($string);

for ($i=0;$i<$duljina;$i++){

echo "[$i]=>".$string[$i]."<br>";

}
```

Da bi znali koliko se for petlja puta mora ponoviti moramo znati duljinu strniga. Nju dobijemo preko funkcije strlen($string) koja vraća integer vrijednost.

Drugi način je korištenjem while petlje.

```
$i=0;

while ($string[$i]){

echo "[$i]=>".$string[$i]."<br>";

$i++;

}
```

While petlji kao uvjet ponavljanja dajemo jedan znak iz stringa. Petlja će se izvršavati sve dok postoji neki znak na danom indeksu, a prekinut će se kada se na danom indeksu više ne nalazi znak. Problem kod ovog načina je taj što će se petlja prestati izvršavati ukoliko se na danom indeksu nalazi 0 (nula) čime ova metoda nije pogodna za sigurno šetanje kroz string.

Primjer 23 e) : Manipuliranje HTML markupa u stringu

```
<hr>

<h3>Manipulacija HTML stringova</h3>

<?

$str_HTML="<p>Ovo je prvi pragraf koji se prikazuje</p>".

"<p>Ovo je drugu paragraf,<br>ali on ima dva reda i jednu <b>boldanu</b> rijec</p>";

echo "<u>HTML koji se ureduje:</u><br>" . $str_HTML;

?>

<hr>

<h3>Zamjena HTML specijalnih znakova</h3>

<?

// zamjena specijalnih znakova

$str_bez_HTML=htmlspecialchars($str_HTML);

echo "<u>HTML sa zamijenjenim specijalnim znakovima:</u><br>" . $str_bez_HTML;

?>

<hr>

<h3>Zamjena nove linije sa &lt;br&gt; tagom</h3>
```

```
<?

$str_nl="Ovo je\nstring koji sadrzi\nvise linija";

echo "Originalni string:<br>$str_nl";

echo "<br><br>String sa zamjennom novih linija sa &lt;br&gt; tagom: <br>";

echo nl2br($str_nl);

?>
```

Rad sa HTML kodom je vrlo bitno pitanje u izradi web aplikacija što se tiče sigurnosti nekih aplikacija poput foruma, formi i sličnih aplikacija koje se oslanjaju na korisnički unos. HTML markupom militantni posjetitelj vrlo lako može natjerati aplikaciju da se ponaša nepredvidivo, čudno i koa rezultat takvog ponašan može doći do probijanja sigurnosti aplikacije ili preopterećenja te time i rušenje servera. Iz tog razloga vrlo je bitno da područja osjetljiva na HTML markup (ona koja prikazuju unesenu sadržaj kao npr. u forumu ili guestbooku) imaju validaciju unosa te da se sav HTML izbaci izunosa ili da se svi HTMl znakovi (< , > , ' , ") zamjene sa kodovima za posebne znakove. Za ovo postoje ugrađene funkcije u PHP-u.

```
// zamjena specijalnih znakova

$str_bez_HTML=htmlspecialchars($str_HTML);

echo "<u>HTML sa zamijenjenim specijalnim znakovima:</u><br>" . $str_bez_HTML;
```

Ovaj fragment koda zamjenjuje sve HTML znakove sa njihovim kodovima za posebne znakove. Za ovo se koristi funkcija htmlspecialchars($str_HTML).

Njoj kao argument dajemo string za kojeg postoji mogućnost da sadrži HTML markup i ona vraća taj isti string za izmijenjenim HTML znakovima. Ova funkcija mijenja slijedeće znakove:

- ' &' (ampersand) postaje '&'
- '"' (dupli navodnik) postaje '"'.
- ''' (Jednostruki navodnik) postaje '''
- '<' (manje) postaje '<'
- '>' (veće) postaje '>'

Druga funkcija koja služi istoj svrsi je htmlentities(). Ona također kao argument prima string i vraća ga sa izmenjenim znakovima. Razlika između ove dvije funkcije je ta što htmlentities()

Izmjenjuje sve znakove koji se mogu promijeneitiu u kodove specijalnih znakova. Ovo znači da će se i znakovi poput č,ć,Č,Đ… mijenjati u njihove reprezentacije u kodu za posebne zakove.

```
$str_nl="Ovo je\nstring koji sadrzi\nvise linija";

echo "Originalni string:<br>$str_nl";

echo "<br><br>String sa zamjennom novih linija sa &lt;br&gt; tagom: <br>";

echo nl2br($str_nl);
```

Druga stvar koja je vrlo česta u gore nevedim aplikacijama je formatiranje unesene vrijednosti. Naime, kada korisnik unosi neki tekst u tekst polje i pritisne enter da bi prenio unos u novi red na mjesto na mjestu na kojem je stisnut enter se nalazi '\n'. Kao što ste već vidjeli u prijašnjim primjerima, ovo se neće vidjeti pri HTMl prikazu ovog unosa u browseru već će se vidjeti tek kada se pogleda source dokumenta. Da bi izbjegli ovo u PHP-u postoji funkcija koja zamjenjuje '\n' sa
 elementom tako da se novi red iz unosa stvarno prikaže kao novi red u pogledu unosa u browseru. Dotična funkcija je nl2br($str_nl)

Primejr 24: Rezanje i lijepljenje stringa

```
<?

$string="Ovo je string kojeg ćemo izrezati na više dijelova";

// rezanje stringa

$arr_izrezani=explode(" ",$string);

foreach($arr_izrezani as $kljuc => $vrijednost){

echo $kljuc . " => " . $vrijednost . "<br>\n";

}

?><hr><?

// lijepljenje stringa iz niza

$arr_stringovi=array("Ovo","je","string","kojeg","ćemo","zalijepiti");

$zalijepljeni=implode(" ",$arr_stringovi);

echo $zalijepljeni;

?>
```

Objašnjenje:

```
// rezanje stringa

$arr_izrezani=explode(" ",$string);
```

Ovaj fragment koda reže zadani string na taj način da podijeli zadani string na fragmente na taj način da je svaki novi fragment dio originalnog stringa između dva znaka koji smo dali kao uvjet razdvajanja u prvom argumentu funkcije. Svi novi fragmenti se spremaju u niz u kojem indeksi odgovaraju rednom broju fragmenta.
Da ovo malo ilustriram na praktičnom primjeru. :

```
// razdjeljivanje e-maila

$email='netko@negdje.com';

$arr_fragemnti=explode('@',$email);

echo 'nick : ' . $arr_fragemnti[0]; // ispisivanje imena iz e-mail adrse

echo 'domena : ' . $arr_fragemnti[1]; // ispisivanje domene e-mail adrese
```

Za fragmentiranje stringa se koristi funkcija
explode($str_uvjet_razdvanjanja,$string_koji_se_razdvaja)

Funkcija prima kao prvi argument znak ili niz znakova koji služe za stvaranje fragmenata. Fragmenti će sadržavati sadržaj danog stringa u drugom argumentu koji se nalazi između dva ponavljanja prvog argumenta. Rezultat funkcije je niz koji sadrži nastale fragmente, a ukoliko se u danom stringu ne nalazi znak ili niz znakova kojim dijelimo string rezultat će biti niz koji sadrži originalni string u cijelosti na svom nultom indeksu.

```
// lijepljenje stringa iz niza

$arr_stringovi=array("Ovo","je","string","kojeg","ćemo","zalijepiti");

$zalijepljeni=implode(" ",$arr_stringovi);

echo $zalijepljeni;
```

Suprotni proces bi bio lijepljenje niza u novi string. Ovo je moguće korištenjem funkcije

```
implode($str_lijepilo,$arr_niz_koji_se_lijepi);
```

Funkcija uzima sve elemente niza iz drugog argumenta i spaja ih tako da zalijepi dva susjedna elementa niza pomoću stringa iz prvog argumenta

Zadaci :

1. Napisati skriptu koja će ispisati sve pozicije pronalaženja nekog stringa ili znaka u stringu te ukupan broj pronalazaka.
2. Napisati funkciju koja će zamijeniti sve hrvatske znakove u njihove nehrvatske zamjene (ć=>c,š=>s…, Ć=>C, Š=>S…)
3. Napisati skriptu koja će boldati sve riječi u stringu koje sadrže slovo 'z'

Rad sa nizovima

Nizovi su osnovni tip elementa koji se najčešće susreće u radi sa bazama podatak. Naime, kada se izvrši upit na bazu vraćeni rezultat je niz. Zato je potrebno znati baratati nizovima i shvatiti njihovu logičku strukturu. Osnovne operacije nad nizovima su šetanje kroz niz, sortiranje niza, pretraživanje niza te lijepljenje niza u string što je već objašnjeno u prethodnom poglavlju.

Primjer 25 a) : Šetanje kroz jednodimenzionalni niz

```
<?

// a) setanje kroz jednodimenzionalni niz.

$arr_valute=array("USD","EUR","GBP","HRK");

foreach ($arr_valute as $kljuc=>$vrijednost)

echo "[$kljuc] => $vrijednost <br>";

?>
```

Objašnjenje:

Ovo je tehnika koja bi već trebala biti poznata jer je korištena kroz cijeli vodič za ispisivanje sadržaja niza. Za šetanje smo koristili foreach petlju koja u svakom svom ponavljanju uzima novi element iz niza. Uzima ih od manjeg indeksa prema većem.

Isti ovaj učinak se može postići na više raznih načina, od kojih je jedan da prvo provjerimo ukupan broj elemenata niza te for petljom prošećemo kroz niz.

```
for($i=0;$i<count($arr_valute);$i++)

echo "[$i] => $arr_valute[$i] <br>";
```

Da bi saznali broj elemenata u nizu koristili smo funkciju count($arr_niz) koja vreća ukupan broj elemenata varijable. Ova tehnika kao što se može vidjeti iz gornjeg primjera radi savršeno za jednodimenz sa numeričkim indeksima.

Što ako se radi o asocijativnom nizu?

U tom slučaju će prva metoda šetanja pomoću foreach petlje raditi savršeno, ali postoje i alternative. Naime, u PHP-u možemo sami određivati koji element niza želimo obrađivati pomoću internog pointera niza. Pointer niza zamislite kao strelicu koja nam pokazuje na kojem se elementu niza nalazimo u danom trenutku. Osnovne funkcije ovakvog načina rada su pomicanje slijedeći – prijašnji te pribavljanje trenutnog indeksa i vrijednosti.

Ovom metodom se šetanje obavlja na slijedeći način.

Primjer 25 b): Šetanje kroz niz korištenjem pointera niza

```
// Asocijativni niz

$arr_assoc=array(

"USD" => 9,

"EUR" => 8,

"GBP" => 10,

"HRK" => 1);

reset($arr_assoc); // resetiranje niza na pocetak

do{

$kljuc=key($arr_assoc); // pribavlajnje klljuca

$vrijednost=current($arr_assoc); // pribavljanje

echo "[$kljuc] => $vrijednost <br>";

} while (next($arr_assoc));
```

Objašnjenje:

Ovaj primjer je univerzalno šetanje kroz jednodimenzionalni niz, iako postoje i druge slične metode. Da bi bili sigurni da će šetanje uvijek početi od prvog unesenog elementa potrebno je prije petlje koja obavlja šetanje vratiti pokazivač niza na početni elemente – resetirati ga. To obavlja funkcija reset($arr_niz).

Druga bitna stvar koju je potrebno uočiti u ovom primjeru je korištenje do..while petlje. Ona je korištena zbog svoje osobine da se njen blok naredbi uvijek izvrši barem jednom prije provjeravanja uvjeta ponavljanja.

U samoj petlji izlučujemo indeks i vrijednost elementa na kojeg trenutno pokazuje pokazivač (pointer) niza. Za svako ponavljanje petlje pokazivač se prenosi za jedno mjesto unaprijed. Redoslijed kojim se pokazivač pomiče je u slučaju asocijativnog niza po redu unosa (prije unesene vrijednosti su na redu prije od kasnije unesenih vrijednosti). Ovo pravilo ne vrijedi nakon sortiranja niza. Nakon njega redoslijed kretanja pokazivača je onakav kako ga je namjestilo sortiranje.

Za izlučivanje trenutnog ključa koristi se funkcija

```
key($arr_niz)
```

a za trenutnu vrijednost se koristi funkcija

```
current($arr_niz)
```

Kao uvijet ponavljanje do … while petlje primjer koristi funkciju next($arr_niz) .

Funkcija prebacuje pokazivač niza na slijedeći element u nizu, a ukoliko ne postoji slijedeći element funkcija vraća False i petlja se prestaje izvršavati.

Suprotna radnja od ove bi bilo ispisivanje elemenata niza od zadnjeg prema prvom. Ovo se izvode tako da se niz prvo namjesti na zadnji element pomoću funkcije end($arr_niz) ,

a u uvjetu ponavljanja petlje se koristi funkcija prev($arr_niz) koja pomiče pokazivač niza za jedno mjesto unazad.

Primjer 26: Šetanje kroz multidimenzionalni niz

Primjer koristi rekurzivno pozivanje funkcije

```
<?

// funkcija za setanje kroz niz

function proseci($niz,$pomak=0){

echo "<pre>";

foreach ($niz as $kljuc => $vrijednost){

for ($i=0;$i<$pomak;$i++)

echo "\t";

if (is_array($vrijednost)){

echo "[$kljuc] => \n";

proseci($vrijednost,$pomak+1);

} else {

echo "[$kljuc] => $vrijednost\n";

}

}

echo "</pre>";

}

// nas niz

$arr_multidim=array(

0 => array("francuz","bijeli","polubijeli","crni","kifla"),
```

```
1 => array(3.5, 5, 4, 3, 0.5),

"kasa" => 500

);

// poziv setanja kriz niz

proseci($arr_multidim);

?>
```

Objašnjenje:

Ovo je malo kompliciraniji primjer zbog korištenja rekurzivnog zvanja funkcije sa default vrijednosti argumenta. Uz to, pri izradi funkcije je uzet u obzir i krajnji izgled rezultata. Razlog ovome je očit. Bez formatiranog ispisa bilo bi vrlo teško odgonetnuti koji ključ (indeks) pripada kojoj dimenziji niza.

Zanimljiva stvar koju možete primijetiti kod ovog multidimenzionalnog niza je da se u PHP-u mogu kombinirati razni tipovi niza (numerički i asocijativni) te da nema predodređenog tipa vrijednosti u nizu za cijeli niz. Vrijednosti unutar jednog niza mogu varirati po volji programera ili potrebi same skripte.

Idemo pogledati funkciju.

```
function proseci($niz,$pomak=0){

echo "<pre>";

foreach ($niz as $kljuc => $vrijednost){

for ($i=0;$i<$pomak;$i++)

echo "\t";

if (is_array($vrijednost)){

echo "[$kljuc] => \n";

proseci($vrijednost,$pomak+1);

} else {

echo "[$kljuc] => $vrijednost\n";

}

}

echo "</pre>";
```

```
}
```

Kao što sam već prije rekao, radi se o funkciji da default argumentom. Riječ je argumentu $pomak. Prije nego što kažem za što njega koristimo idem prvo pokušati ukratko opisati što radi ova funkcija.

Očito je da funkcija kao ulazni parametar prima niz, nebitno koje dimenzije, te stvara formatirani ispis tog niza. Za formatiranje su korišteni tzv. escape znakovi \n za novi red i \t za tabulatore. Iz tog razloga je potrebno sav ispis umetnuti između <pre> … </pre> tagova jer bi u protivnom cijeli ispis bio u jednom redu. Ispis se formatira na taj način da se svaka nova dimenzija niza ispisuje za jedno 'tabulatorsko' mjesto udesno. I ovdje dolazi defaultni argument na djelo. Pomoću njega govorimo funkciji u kojoj se dimenziji nalazimo tj. za koliko mjesta treba pomaknuti ispis udesno. Bez ovog pomicanja bilo bi teško odrediti pripadajuće dimenzije i indekse pošto se ponavljaju za svaku dimenziju. Istina, funkcija se mogla riješiti i bez defaultnog argumenta, ali radi jednostavnosti, razumljivosti i čistine krajnjeg koda je ipak upotrijebljen. Ovako se u glavnom programu funkciji daje samo niz, a funkcija se sama brine za formatiranje ispisa.

Sada kada je svrha i funkcija funkcije opisana idemo se pozabaviti njenom logikom. Pošto funkcija ispisuje niz logično je da ćemo se prošetati nizom. Najjednostavniji način za to je foreach petljom. Slijedeća stvar koju je potrebno napraviti je utvrditi kojeg je tipa trenutni element niza. Nas zanimaju dva tipa. Niz i neniz.

Ukoliko se radi o nizu njega je potrebno ispisati na isti način kao i njegovog roditelja, tj. dimenziju u kojoj se on nalazi. Ukoliko se ne radi o nizu želimo ispisati vrijednost na ekran.

Za ispitivanje tipa varijable postoji asortiman funkcija za tu namjenu. Nama je potreban funkcija is_array($varijabla) koja, kako joj ime govori, ispituje da li je zadana varijabla niz. Ukoliko je niz, rezultat je true, a u suprotnom je rezultat false.

```
if (is_array($vrijednost)){

echo "[$kljuc] => \n";

proseci($vrijednost,$pomak+1);

}
```

Ovaj blok koda se izvršava ukoliko je trenutna vrijednost niz. On ispisuje trenutni ključ (indeks) na kojem se nalazimo i prenosi ispis u novi red. Slijedeći korak je pozivanje ove iste funkcije, ali ovog puta joj dajemo drugi argument.

Sada je bitno razumjeti za što služi drugi, default argument. Pomoću njega formatiramo ispis. On označava pomak udesno naspram trenutnog pomaka. Pomak se realizira for petljom prije bilo kakvog ispisa u foreach petlji. Ovime osiguravamo da će svaki element dimenzije, ukoliko nije niz biti ispisan na istom pomaku, ili ukoliko je niz njegov indeks će biti u tom pomaku, a njegovi elementi će opet biti pomaknuti za jedno mjesto udesno.

```
for ($i=0;$i<$pomak;$i++)

echo "\t";
```

Drugim riječima, argumentom $pomak govorimo skripti koliko se duboko nalazimo u nizu.

```
} else {

echo "[$kljuc] => $vrijednost\n";

}
```

U slučaju da element niza nije i sam niz, ispisuje se njegova vrijednost i indeks te se ispis prenosi u novi red.

Primjer 27: Pretraživanje niza

```
Niz :<br>

<?

$arr_mailovi=array(

"kkondza@hotmail.com",

"kongi@programiranje.net",

"kkondza@hotmail.com",

"marko@radionica.com",

"ines@radionica.com",

"mario@hinet.hr",

"vladimir@inet.hr"

);

// ispisivanje niza

foreach ($arr_mailovi as $mail){

echo $mail ." <br>";

}

$str_trazeni="kkondza@hotmail.com";

?>

<hr>

String koji se traži :<?=$str_trazeni?><br>

<?

// traženje vrijednosti u nizu i vracanje indeksa na kojem se nalazi vrijednost
```

```
if ($nadeni=array_search ($str_trazeni,$arr_mailovi) or $nadeni===0){

echo "Pronađen je element u nizu na indeksu $nadeni => $arr_mailovi[$nadeni]<br>";

} else {

echo "Traženi string ne postoji u nizu<br>";

}

?><hr><?

//pretraživanje cijelog niza i ispis svih indeksa - FALIENO

while ($nadeni=array_search ($str_trazeni,$arr_mailovi) or $nadeni===0){

echo "Pronađeno na poziciji : $nadeni => $arr_mailovi[$nadeni]<br>";

unset ($arr_mailovi[$nadeni]);

}

?>
```

Objašnjenje:

U ovom primjeru cilj nam je izdvojiti zadani e-mail iz niza koji sadrži popis e-mailova. E-mail koji tražimo se zadaje u varijablu $str_trazeni.
Prvi način pretraživanja pronalazi samo prvo pojavljivanje u nizu, a ostala su zanemarena.

```
// traženje vrijednosti u nizu i vracanje indeksa na kojem se nalazi vrijednost

if ($nadeni=array_search ($str_trazeni,$arr_mailovi) or $nadeni===0){

echo "Prona?en je element u nizu na indeksu $nadeni => $arr_mailovi[$nadeni]<br>";

} else {

echo "Traženi string ne postoji u nizu<br>";

}
```

Za traženje vrijednosti u nizu koristi se funkcija array_search($trazeno,$niz_za_pretraziti):

Funkcija prima dva ovaezna argumenta. Prvi argument je vrijednost koja se traži, a drugi je niz u kojem se traži vrijednost. Postoji i treći, neobavezni, argument koji može biti true ili false. On služi ukoliko se želi usporediti i tip podataka. Recimo $int_broj=111 i $str_broj='111' u slučaju da je treći argument postavljen na true ne bi rezultiralo pronalaženjem vrijednosti u nizu.

Funkcija će vratiti indeks na kojem je pronađena vrijednost, a ukoliko tražena vrijednost ne postoji vraća se False.

Primijetite uvjet koji se nalazi u if strukturi :

```
($nadeni=array_search ($str_trazeni,$arr_mailovi) or $nadeni===0)
```

Prvo u varijablu $nadeni spremamo rezultat traženja. U nju će se spremiti indeks na kojem je pronađena vrijednost ili False. Drugi uvjet provjerava da li je ta vrijednost slučajno broj 0. Ovo je potrebno zato što ukoliko se tražena vrijednost nalazi na indeksu 0 if struktuta taj rezultat tretira kao False i obavlja se else grana if-a. Znači da bi se bez drugog uvjeta nulti elemnt niza zanemarivao pri pretraživanju. Uvjet $nadeni===0 provjerava da li je rezultat funkcije broj nula ili vrijednost false. Ukoliko je broj nula ipak će se obaviti true grana if-a i ovime je zagarantirana točnost pretraživanja.

Ali naš niz sadrži dvije iste vrijednosti na indeksima 0 i 2. Prijašnji kod neće pretražiti cijeli niz već će stati nakon prvog pronalaženja.

Da bi pretražili cijeli niz očito je da je potrebno koristiti neku ponavljačku petlju.

```
//pretraživanje cijelog niza i ispis svih indeksa – FALICNO

while ($nadeni=array_search ($str_trazeni,$arr_mailovi) or $nadeni===0){

echo "Prona?eno na poziciji : $nadeni => $arr_mailovi[$nadeni]<br>";

unset ($arr_mailovi[$nadeni]);

}
```

Za ovaj slučaj while petlja je vrlo praktična. Ona će se ponavljati sve dok u nizu postoji tražena vrijednost. Ako se pitate kako se petlja neće ponavljati u beskonačnost zato što je u prošlom primjeru rečeno da se ona zaustavlja nakon pronalaska prvog pojavljivanja, dobro se pitate. Naime, petlja će uvijek pronaći samo prvo pojavljivanje i vrtiti će se u beskonačnost. Da bi izbjegli ovo ponavljanje potrebno je nakon što pronađemo traženu vrijednost nju i izbaciti iz niza tako da slijedeće ponavljanje petlje pretražuje ostatak niza.

Element iz niza se izbacuje unset($var) funkcijom.

Funkcija unset() uništava danu varijablu. Nju se više ne može kasnije koristiti u skripti. Ukoliko je potrebno izbaciti samo jedan element niza funkciji dajemo taj niz sa željenim indeksom ($arr_niz[16]) i on će biti izbačen.

U primjeru se element izbacuje ovim redom

```
unset ($arr_mailovi[$nadeni]);
```

Ova metoda je falična zato što nakon što pretražimo niz u njemu se neće nalaziti pronađene vrijednsoti, pa ih se time ne može koristiti kasnije u skripti. Ovo se može izbjeći korištenjem funkcije koja će obaviti traženje što je objašnjeno u slijedećem primjeru

Primjer 27b : Pretraživanje niza sa više pojavljivanja tražene vrijednosti

```
<?

// traženje ponovljenih vrijednosti pomoau funkcije

function trazi($vrijednost,$niz){

while ($nadeni=array_search ($vrijednost,$niz) or $nadeni===0){

echo "Prona?eno na poziciji : $nadeni => $niz[$nadeni]<br>";

unset ($niz[$nadeni]);

}

}

// provjeranje postojanja tražene vrijednosti u nizu

if (in_array($str_trazeni,$arr_mailovi)){

trazi($str_trazeni,$arr_mailovi);

} else {

echo "U nizu ne postoji trazena vrijednost";

}

?>
```

Objašnjenje:

U ovom primjeru prije ispisivanja prvo provjeravamo da li u nizu postoje tražene vrijednosti. Postojanje vrijednosti u nizu se provjerava funkcijom in_array($str_trazeni,$str_niz_pretrazivanje).

Funkcija vraća True ukoliko vrijednost postoji,a false u suprotnom. Struktura argumenata funckije je identična i array_search() funkciji, pa i ova također prima treću, neobavezni, argument za uspoređivanje tipa podataka.

Prednost ove metode nad one bez korištenja funkcije je ta što kada predamo niz funkciji ona stvara lokalne (funkcijske) varijable za svoje argumente, te izbacivanje elemenata iz niza ne utječe na sadržaj predanog niza iz glavnog programa te taj niz možemo opet naknadno koristiti u skripti.

Primjer 28: Sortiranje niza

Postoji mnogo raznih algoritama za sortiranje niza i u ovom primjeru će biti objašnjeni samo neki od njih. Isto tako PHP je opremljen već gotovim funkcijama za sortiranje po indexu ili vrijednosti jednodimenzionalnih i multidimenzionalnih nizova.

```
<?
$arr_niz=array(
0=>10,
1=>15,
2=>9,
3=>19,
4=>13,
5=>15,
6=>99,
7=>74
);
// sortiranje niza
asort($arr_niz);
print_r($arr_niz);
?><hr><?
arsort($arr_niz);
print_r($arr_niz);
?><hr><?
ksort($arr_niz);
print_r($arr_niz);
?><hr><?
krsort($arr_niz);
print_r($arr_niz);
?>
```

U ovom vrlo jednostavnom primjeru se zadani niz sortira po raznim uvjetima.

U prvom slučaju niz je sortiran po vrijednostima niza uzlazno. Za sortiranje je korištena funkcija asort($arr_niz)

Njoj se kao argument daje niz koji se sortira. Funkcija ne vraća vrijednost, već samo sortira niz, i nakon njenog izvršenja je zadani niz sortiran i takvom mu pristupamo kasnije u skripti. Sortiranje se obavlja tako da se elementi niza sortiraju uzlazno. Ono što je bitno kod ove funkcije je da se održava odnos indeks => vrijednost, tako da će nakon sortiranja vrijednost koja se nalazila na indeksu 3 i dalje biti na tom indeksu, samo će njena pozicija u samom nizu možda biti drugačija.

U drugom slučaju niz se sortira silazno (od najveće prema najmanjoj vrijednosti). To se radi pomoću funkcije arsort($arr_niz). Ova funkcija je identična asort() funkciji po svom osobinama osim po redoslijedu elemenata nakon sortiranja.

U trećem slučaju niz se sortira po ključu. Znači da će ključ (indeks) sa manjom vrijednosti biti prije u nizu pošto funkcija ksort($arr_niz) sortira niz po indeksu uzlazno. Odnos indeks => vrijednost je zadržan nakon sortiranja.

Zadnji slučaj sortira niz po indesku, ali silazno. Tu radnju obavlja funkcija krsort($arr_niz). Ona je identična funkciji ksort(), samo što sortira niz obrnutim redosljedom

Zadaci:

1.Napisati skriptu koja će pretraživati postojanje elemenata jednog niza u drugom nizu. Skripta treba izbaciti sve pronađene vrijednosti u drugom nizu, a u prvom nizu treba naznačiti da je vrijednost pronađena te ispisati oba niza.

2.Napisati funkciju koja će pretraživati multidimenzionalni niz i vratit će točan indeks svih dimenzija do mjesta na kojem je pronađena vrijednost. Vraćena vrijednost je također niz u kojem svaki indeks označava dubinu (dimeziju). Npr. za pronađenu vrijednost na dimenziji $arr_niz[0][2][5] rezultat treba biti $arr_rezultat[0]=0, $arr_rezultat[1]=2, $arr_rezultat[2]=5.

3.Napisati skriptu koja će ispisati samo one elemente koji u sebi sadrže niz nakova ABC

Višenamjenske stranice

U ovom članku je objašnjenja logika višenamjenskih stranica i u konkretnim primjerima su detaljno objašnjene njihove različite implementacije.

Prije samih primjera potrebno je postaviti i objasniti problem i sam pojam višenamjenskih stranica.

U izradi serverskih aplikacija, nebitno u kojem jeziku su one izvedene, je ponekada praktičnije koristiti jedan fizički dokument za obavljanja različitih logički povezanih operacija. Ovakve situacije se najčešće javljaju u portalskim aplikacija koji se sastoje od više sekcija i servisa. Korištenjem višenamjenskih stranica je moguće fizički separirati svaki odjeljak ili servis cjelokupne aplikacije u posebne datoteke (skripte) koje imaju specijaliziranu namjenu. U konkretnom slučaju, jedna višenamjenska stranica bi se koristila za listanje, pretraživanje i prikazivanje vijesti iz arhive vijesti, a druga recimo za servis e-razglednica ili forum.

Tehnički gledano, višenamjenske stranice nisu ništa drugo nego obavljanje drugog niza instrukcija (operacija) za različite situacije koje se određuju pomoću nekog kontrolnog uvjeta koji može biti kontrolna riječ, koja je za svaku situaciju drugačija, ili kontrolna varijabla koja za svaku situaciju ima drugu vrijednost.

Uobičajena praksa prosljeđivanja kontrolnog uvjeta je putem komandne linije (dijelu URL-a iza znaka ?).
Višenamjenska stranica izvedena korištenjem kontrolne riječi

Ovo je jedna od metoda implementacije višenamjenskih stranica. Kontrolna riječ je fragment komandne linije (ASP-ovci ovo znaju kao Querry String) kojem nije pridružena nikakva vrijednost (nema niti znak =) i odvojen je od ostalih parametara komandne linije (uobičajeno razdvajanje za PHP je pomoću znaka &).

Evo ako bi izgledao primjer:

http://www.webmajstori.net/visenamjenska.php? webmajstori

Znači, u gornjem primjeru kontrolna riječ je 'webmajstori'

Evo i skripte.

```
<?

switch(@$_SERVER["QUERY_STRING"]){

case "kreso":

echo "Pozdrav majstore";

break;

case "webmajstori":

echo "To su isto majstori";

break;

default:

echo "A tko si ti?";

}



echo "<hr>";

echo "<p>Sadržaj query stringa<br>";

echo $_SERVER["QUERY_STRING"];

echo "</p>";

echo "<hr>";

echo "<p>Provjera postojanja varijable putem isset funkcije<br>";

if (isset($_GET["kreso"])){

echo "varijabla kreso postoji<br>";
```

```
}else{

echo "varijabla kreso ne postoji<br>";

}

if (isset($_GET["webmajstori"])){

echo "varijabla webmajstori postoji<br>";

}else{

echo "varijabla webmajstori ne postoji<br>";

}

echo "</p>";



?>
```

Analiza:

Za gore navedeni primjer rezultat bi izgledao ovako :

```


To su isto majstori

Sadržaj query stringa
webmajstori

Provjera postojanja varijable putem isset funkcije
varijabla kreso ne postoji
varijabla webmajstori ne postoji
```

Idemo sada proletiti kroz sam kod.

Prvi dio skripte obavlja neke radnje na osnovi zadane kontrolne riječi i tu zapravo izvodimo višestruku namjenu ove skripte.

```
switch(@$_SERVER["QUERY_STRING"]){

case "kreso":

echo "Pozdrav majstore";

break;

case "webmajstori":
```

```
echo "To su isto majstori";

break;

default:

echo "A tko si ti?";

}
```

Mislim da nije potrebno objašnjavati ovu switch strukturu, ali da bi bili sigurni idem ju ipak objasniti. Switch provjerava tekst koji se nalazi u komandnoj liniji i uspoređuje ga sa vrijednostima u case granama. Ukoliko je taj string jednak jednoj od ponuđenih vrijednosti ispisuje se odgovarajuća poruka, a ukoliko ne odgovara niti jednoj od tih vrijednosti ili nema vrijednosti vraća se poruka 'A tko si ti?' tj. default grana switch strukture.

Zašto je potrebno pristupati kontrolnoj riječi na gornji način (preko (@$_SERVER["QUERY_STRING"])?
Razlog ovome je zapravo vrlo jednostavan. Prosljeđivanjem kontrolne riječi u komandnoj liniji bez pridružene vrijednosti i znaka = ne definira nikakvu varijablu, pa time nije moguće ispitivati njenu vrijednost ili njeno postojanje.

Umjesto toga pristupamo cijelom tekstu iza znaka ? (tj. query stringu) koji se nalazi u varijabli $_SERVER["QUERY_STRING"]

Da prosljeđivanjem ključne riječi PHP ne definira varijablu dokazujem slijedečim kodom u skripti.

echo "<p>Provjera postojanja varijable putem isset funkcije
";

```
if (isset($_GET["kreso"])){

echo "varijabla kreso postoji<br>";

}else{

echo "varijabla kreso ne postoji<br>";

}

if (isset($_GET["webmajstori"])){

echo "varijabla webmajstori postoji<br>";

}else{

echo "varijabla webmajstori ne postoji<br>";

}

echo "</p>";
```

Ovaj dio koda vraća za gornji primjer :

varijabla kreso ne postoji
varijabla webmajstori ne postoji

Korištenje kontrolne riječi otvara više novih problema. Naime, ukoliko želimo skripti proslijediti preko komandne linije i neku varijablu koja nosi isto ime kao i jedna od naših kontrolnih riječi, ili samoj kontrolnoj riječi pridružimo neku vrijednost uvijek će se obavljati default grana switch strukture pošto cijela komandna linija u tim slučajevima (pogledaj dole) neće odgovarati ponuđenim mogućim stanjima u svakom od case odjeljaka (grana) switch strukture.

http://www.webmajstori.net/visenamjenska.php?
webmajstori=123

http://www.webmajstori.net/visenamjenska.php?
webmajstori&webmajstori=123

http://www.webmajstori.net/visenamjenska.php?
webmajstori&kreso=123

Kako to izbjeći?

Dobro pitanje. Ono što moramo u tom slučaju (za podsjetnik – imati još dodatne argumente u query stringu) napraviti je postaviti neka pravila. Ta su da se kontrolna riječ uvijek mora nalaziti prva iz znaka ? u URL-u. Nakon uvođenja ovog pravila možemo izbjeći gore navedeni problem sa slijedećim kodom :

```
/*

Pribavljanje ključne riječi

znakom @ izbjegavamo prikaz greške ukoliko je query string prazan

*/

$arr_komandna_linija=@explode("&", $_SERVER["QUERY_STRING"]);

$str_komandna_rijec=$arr_komandna_linija[0];

echo "<br>Vraćena vrijednost nakon provjere je <br>";

switch(@$str_komandna_rijec){

case "kreso":

echo "Pozdrav majstore";

break;

case "webmajstori":

echo "To su isto majstori";
```

```
break;

default:

echo "A tko si ti?";

}

echo '</p>';



echo "<hr>";

echo "<p>Sadržaj query stringa<br>";

echo $_SERVER["QUERY_STRING"];

echo "</p>";

echo "<hr>";

echo "<p>Provjera postojanja varijable putem isset funkcije<br>";

if (isset($_GET["kreso"])){

echo "varijabla kreso postoji<br>";

}else{

echo "varijabla kreso ne postoji<br>";

}

if (isset($_GET["webmajstori"])){

echo "varijabla webmajstori postoji<br>";

}else{

echo "varijabla webmajstori ne postoji<br>";

}

echo "</p>";



Idemo analizirati ovaj kod.

$arr_komandna_linija=@explode$_SERVER["QUERY_STRING"]);
```

Ovim kodom u varijablu $arr_komandna_linija spremao sav sadržaj komandne linije tako da svaki element niza sadrži string vrijednost između dva znaka &. Ovime će u nultom elementu toga niza biti naša kontrolna riječ koju kasnije možemo provjeravati, a u isto vrijeme možemo imati i dodatne varijable za kasnije korištenje u skripti. Ukoliko se u query stringu nalazi samo kontrolna riječ bez znaka & u nultom elementu će se i dalje nalaziti kontrolna riječ. Ukoliko je query string prazan znakom @ smo onemogućili javljanje greške i u switchu će se obaviti default grana, tj. vratiti će se poruka 'A tko si ti?'.

Za izlučivanje smo koristili funkciju

```
explode($str_znak_odvajanja , $str_koji_zelimo_razdvojiti)
```

Ova funkcija rastavlja $str_koji_zelimo_odvojiti tako da u niz vraća dijelove tog stringa koji su u njemu bili odvojeni sa $str_znak_odvajanja.

Da malo ovo ilustriram.

Imamo string :

Maja,Ivana,Matea,Branka,Ana

I želimo ga razdvojiti tako da je svako ime odvojeno tj. želimo imati ovo:

```
Maja
Ivana
Matea
Branka
Ana
```

Za to koristimo explode funkciju na slijedeći način:

```
$str_imena=' Maja,Ivana,Matea,Branka,Ana';

$arr_imena=explode(',',$str_imena);
```

Sada se u $arr_imena nalaze slijedeće vrijednosti

```
$arr_imena[0]='Maja'

$arr_imena[1]='Ivana'

…
```

Vratimo se našoj skripti.

Ostatak koda u njoj funkcionira isto kao i prvobitnom primjeru.

Sada će ako samo dodamo ovaj kod na kraj skripte

http://www.webmajstori.net/visenamjenska.php?webmajstori&kreso=123

vratiti:

Vraćena vrijednost nakon provjere je
To su isto majstori

Sadržaj query stringa
webmajstori&kreso=123

Provjera postojanja varijable putem isset funkcije
varijabla kreso postoji
varijabla webmajstori ne postoji

Višenamjenska stranica korištenjem kontrolne varijable

Ovo je jednostavnija metoda implementacije višenamjenskih stranica. Ideja je koristiti varijablu koja se daje skripti preko komandne linije. Ova varijabla (zapravo njena vrijednost) određuje što će se prikazivati na stranici. Kao dodatno pravilo koje se uvodi radi jednostavnosti same skripte i sigurnosti aplikacije je određivanje akcije ukoliko kontrolna varijabla ne postoji. U ovoj skripti u toj situaciji se prikazuje nešto kao početna stranica na kojoj se nalaze linkovi na druge moguće slučajeve. Problem (ili prednost) kod ovakvog rješenja je da će se ta početna stanica prikazivati i u slučaju da korisnik sam promijeni vrijednost kontrolne varijable u nepostojeću (nepodržanu) vrijednost ili ju izostavi u cijelosti.

Evo skripte:

```
<?

// visenamjensak stranica koristenjem kontrolne varijable

// kontrolna varijabla : $kon

switch (@$kon){

case 1:

echo "Gledate stranicu 1<br>";

echo "<a href='$PHP_SELF'>Povratak na poeetnu stranicu</a>";

break;

case 2:
```

```
echo "Gledate stranicu 2<br>";

echo "<a href='$PHP_SELF?kon=33'>Povratak na poeetnu stranicu</a>";

break;

case 3:

echo "Gledate stranicu 3<br>";

echo "<a href='$PHP_SELF'>Povratak na poeetnu stranicu</a>";

break;

default:

echo "<h2>Poeetna stranica </h2>";

echo "<p>Odaberite jednu od stranica<br>";

echo "<ul><li><a href='$PHP_SELF?kon=1'>Stranica 1</a></li><li><a
href='$PHP_SELF?kon=2'>Stranica 2</a></li><li><a href='$PHP_SELF?kon=3'>Stranica
3</a></li><ul>";

echo "</p>";

}

?>
```

Rezultat skripte ovisi o stanju kontrolne varijable, ali ilustracije radi promotrite ove situacije

Za

http://www.host.com/skripta.php

Rezultat izgleda:

```
Početna stranica

Odaberite jednu od stranica

Stranica 1

Stranica 2

Stranica 3
```

Za

http://www.host.com/skripta.php?kon=1

Rezultat izgleda:

```
Gledate stranicu 1
Povratak na početnu stranicu
```

Za

http://www.host.com/skripta.php?kon=33

Rezultat izgleda:

```
Početna stranica

Odaberite jednu od stranica

Stranica 1

Stranica 2

Stranica 3
```

Idemo analizirati kod:

Cijela skripta se zapravo sastoji od jednog 'velikog' switcha koji odlučuje što se obavlja ovisno o kontrolnoj varijabli. Kontrolna varijabla u ovoj skripti je $kon.

Ova varijabla ne mora postojati da bi skripta radila. Ovo je moguće zato što je u switchu korišten znak @ koji isključuje prikaz greške koja bi se prikazala u situaciji da kontrolna varijabla ne postoji. Ovako će se u slučaju da ona ne postoji obaviti default grana switcha. Ista grana će se obaviti i ako trenutna vrijednost kontrolne varijable nije ponuđena niti u jednom caseu.

Ostatak skripte objašnjava sam sebe. U defaultu se prikazuje popis svih ostalih mogućih odabira za ovu skriptu, a u svakom caseu se obavljaju radnje specifične za taj odabir.

U svakom case je također ponuđen povratak na početnu stranicu. Linkovi se razlikuju u svakom od njih čisto ilustracije radi.

Mogući povratci su :

```
<a href='$PHP_SELF'>Povratak na pocetnu stranicu</a>
```

Ovim linkom se stranici ne daje kontrolna varijabla i iz već objašnjenih razloga se obavlja default grana switcha. Ovo je ujedno i najbolji način vraćanja korisnika na početnu stranicu.

```
<a href='$PHP_SELF?kon=33'>Povratak na poeetnu stranicu</a>
```

U ovom primjeru se skripti daje nepostojeća (neponuđena) vrijednost (stanje) kontrolne varijable. Pošto vrijednost ne postoji obavlja se default grana switcha. U ovoj situaciji se može postaviti bilo koja neponuđena vrijednost kontrolne varijable. Ova metoda vraćanja možda nije najbolji izbor. Naime, što ako se tokom vremena poveća broj mogućih stanja kontrolne varijable? Bit ćete prisiljeni mijenjati cijelu skriptu.

Moguća stanja kontrolne varijable mogu biti bilo što. U ovom primjeru su korišteni cijeli brojevi radi jednostavnosti. Naime, mogućnost greške je mnogo veća ukoliko su moguća stanja neki veliki stringovi.

Normalno, odabir mogućih vrijednosti ostavljam vama, ali pri odabiru je potrebno uzeti u obzir neke sigurnosne konsideracije. Ukoliko se koriste integer vrijednosti militantni korisnik uvijek može vrlo lako pogoditi stanje, dok će kod korištenja stringova imati malo više problema oko pogađanja stanja. U svakom slučaju, odabir je na vama.

Zadaci:

1.Ispraviti formu iz primjera 31. Potrebno je dodati validaciju (ime i prezime moraju sadržavati samo tekst, email mora sadržavati znak @) te popraviti bug oko unosa 0 u bilo koje polje

2.Napraviti formu koja će od korisnika tražiti da unese svoje ime, prezime, mail, URL svog homepagea te neki tekst. Potrebno je obaviti sve validacije polja i nakon nih ukoliko su uspješno obavljene ispisati tekst iz tekst polja tako da svaki novi red u polju bude zamijenjen
 tagom te da se niz nakova :) zamijeni sa riječi SMAJLI.
Dodatno možete omogućiti da se uneseni tekst između [vazno] neki tekst [/vazno] ispiše boldano.

Session managment

Sessioni su vrlo bitna stvar pri izradi Internet aplikacija. Prvo da malo razjasnim sto su to sessioni, kako funkcioniraju te čemu služe.

U izradi Internet aplikacija postoji velika potreba za nekakvim mehanizmom za pamćenje informacija vezanih uz jedan posjet korisnika našoj aplikaciji (siteu). Taj mehanizam su sessioni. Iz ovoga se može izvući i zaključak što je session. On bi bio jedan posjet siteu jednog korisnika. Session traje sve dok je korisnik na siteu. Kada korisnik prestane otvarati stranice na siteu PHP ima kojim nakon zadanog vremena (vrijeme se zadaje u php.ini-u) briše sve informacije koje su spremljene tokom sessiona na hard disc servera. Ovako se štedi na prostoru.

Da vam malo ilustriram kako funkcioniraju sessionu pokušat ću vam ispričati jednu malu priču.

Korisnik X dolazi na site. PHP mu automatski pridružuje SID (session ID). Pomoću tog id-a se naš korisnik X kasnije identificira kada posjeti neku drugu stranicu na siteu. Kada korisnik dođe na site, skripta automatski stvori session varijablu u koju će se spremati broj posjećenih stranica u ovom sesionu. Ta varijabal se namješta na 0. Sada korisnik odabire svoju omiljenu temu na našem siteu i odlazi na stranicu na kojoj se nalazi više informacija o toj temi. Normalno, mi žalimo dodati u našu session varijablu i ovaj posjet. Skripta to i čini, te mu prikazuje traženi sadržaj. Korisnik čita tekst koji je tražio na prvoj stranici i polako se počinje živcirati jer se ne slaže sa fundamentalnim

pitanjima koje je autor teksta koji čita predstavio vrlo neprofesionalno i nepromišljeno. Pred kraj teksta korisnik već počinje nervozno micati miš po ekranu sa krajnjim odredištem na svojim favoritesima i odabire neki site iz njih koji mu neće prouzročiti toliko stresa koliko naš site te odlazi sa našeg sitea.

Sada je ostatak session managmenta na samom PHP-u. Pošto korisnik X koji je identificiran sa svojim SID-om ne radi nove zahtjeve na našem siteu (pošto nam je skinuo sve svece sa neba i pobjegao od neprofesionalnosti našeg autora teksta) PHP zna da se session 'gasi' nakon 30 minuta nakon što je korisnik otišao sa našeg sitea. Sada PHP strpljivo čeka da to vrijeme prođe da može obrisati varijable iz svog temp direktorija koje je skripta stvorila za korisnika X. Ali nekim čudom se korisnik vraća nakon 15 minuta, sada još ljući nego kada je otišao, i traži način kako da nam da do znanja da smo potpuno promašili člankom koji smo objavili na našem siteu. Vraća se na stranicu na kojoj je čitao tekst i primjećuje na dnu stranice formu u koju može unijeti svoj komentar. U trenutku kada se korisnik X vratio na site, sve session informacije (broj posjećenih stranica) i dalje postoje i umjesto da se stvara nova session varijabla sa brojem posjećenih stranica sa nulom za vrijednost, brojanje se nastavlja tamo gdje je stalo kada je korisnik otišao skidajući nam svece sa neba.

Iako je ovo vrlo jednostavna situacija (samo se broje posjećene stranice) ipak znamo da se ne radi o novom korisniku, već o starom korisniku koji nam je pomogao.

Sada dolazimo do pitanja za što koristiti session-e. U gornjoj situaciji smo ga koristili za brojanje posjećenih stranica korisnika X. To je jedan od mogućih namjena. Druga bi recimo bila brojanje ukupnih posjeta našem siteu koji bi bili jednaki broju stvorenih sessiona. U toj situaciji bi pri stvaranju session varijable negdje u bazu ili neki flat file (npr. txt file je flat file) spremili novi broj posjeta siteu.

Još jedna korisna stvar kojas emože 'čuvati' u session varijablama su korisničke informacije poput nicknamea, userID-a i sličnih stvari, zatim sadržaj košarice u e-shop aplikaciji.

Generalno bi se dalo reći da u session varijable spremamo informacije koje tiču korisnika i njegovom posjetu siteu. Te informacije ne moraju, ali mogu biti od velike važnosti za cijelu aplikaciju, ali u 99% situacija one služe samo za olakšan rad tom korisniku (zamislite da korisnik mora negdje na papir zapisivati sve artikle koje želi kupiti na našem siteu).

A sada idemo pogledati aplikaciju iz gornje pričice.

Primjer 32 : Brojanje posjećenih stranica

Primjer koristi višenamjensku strancu!

```
<? ob_start() ;// output buffering

session_start(); // pokreatnje sessiona

// namještanje session varijable

if (session_is_registered("posjeta")){

$posjeta++;

} else {

$posjeta=1;
```

```
}

// registriranje session varijable

session_register("posjeta");

?>

<html>

<head>

<title>Session managment</title>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">

</head>

<body>

<?

// višenamjenska stranica

// kontrolna varijabla $k

/* STANJA $k

1 - ispsi elanka koji je jako stresan

2 - stranica za unos i gledanje komentara

default - poeetna stranica sitea

*/

switch (@$k){

case 1:

?>

<h1>Elanak</h1>

<p>Ovo je taj elanak od kojeg se eovjeku diže kosa na glavi</p>

<p>Nažalost morali smo ga cenzurirati zbog velikog broja pritužbi koje su dospjele
na naš mail</p>

<form action="<?=$PHP_SELF?>?k=2" method="post" name="forma" id="forma">

<p>Vaše ime:

<input type="text" name="ime">
```

```
<br>

Vaš mail

<input type="text" name="textfield">

<br>

Komentar:

<textarea name="komentar" cols="25" rows="10" id="komentar"></textarea>

<br>

<input type="submit" name="Submit" value="Dodaj komentar">

</p>

</form>

<?

break;

case 2:

// obarda forme i prikaz komentara

if (@$_POST["Submit"]){

// fali validacija forme!!!

echo "Vase ime : <b>" . $_POST["ime"] . "</b><br>";

echo "Vaš komentar <pre>\n" . $_POST["komentar"] . "</pre><br>";

?>

<a href="<?=$PHP_SELF?>">Povratak na poeetnu stranicu</a>

<?

} else {

// prikaz svih ostalih komentara

// potrebno je negdje spremiti komentare, npr u bazu

?>

<p>Nažalost nismo još uvijek u moguenosti prikazati sve komenatre</p>

<?

}
```

```
break;

default:

?>

<p>DObrodošli na ovaj potono otkvaeen site</p>

<p><a href="<?=$PHP_SELF?>?k=1">Pogledaj elanak od kojeg ae ti se dignuti kosa

na glavi </a>

<?

}

echo "<p>Do sada ste posjetili $posjeta stranica na ovom siteu</p>";

ob_end_flush();

?>

</p>

</body>

</html>
```

Objašnjenje:

Postoji jedna vrlo bitna stvar kod rada sa sessionima. Sve operacije vezane uz njih se moraju obaviti prije nego što se bilo kakav sadržaj pošalje korisniku. Ovo s emože osigurati na više načina.

Prvi je da se te operacije stave na sam početak skripte. U tom slučaju prije tih operacija ne smije biti nikakav kod, HTML ili prazan red. Ovo je u većini situacija vrlo teško postići. Gotovo nemoguće.

Drugi način za onemogućiti slanje bilo kakvog sadržaja korisniku je korištenjem output buffering funkcija. Output buffering je kontroliranje slanja generiranog HTML koda korisniku. Znači da na samom početku skripte možemo isključiti slanje outputa PHP koda (ili drugim riječia uključiti output buffering). Ovime osiguravamo da se korisniku neće ništa poslati prije nego li mi kažemo da se šalje. Ovom metodom se session operaciej mogu obavljati bilo gdje na stranici.

```
<? ob_start() ;// output buffering
```

Ovime je uključen output buffering i korisniku se neće slati nikakav sadržaj prije pojavljivanja

```
ob_end_flush();
```

Koji se nalazi pri kraju skripte.

Sada kada smo to riješili možemo se pozabaviti logikom koja je potrebna za brojanje posjećenih stranica. Praktički, ono što želimo napraviti je pri svakom otvaranju skripte uvećati neku varijablu za jedan te zapamtiti vrijednost te varijable za kasnije korištenje kroz session.

Problem koji se otvara je kako znamo da li je trenutno otvaranje skripte (stranice) prvo, ili je korisnik već posjetio neku stranicu na siteu. Iz tog razloga prvo provjeravamo da li postoji varijabla u koju se sprema broj posjećenih stranica te ukoliko postoji njena vrijednost se uvećava, a ukoliko ne postoji stvaramo tu varijablu i namještamo joj početnu vrijednost na 1 pošto je korisnik posjetio jednu (ovu koju gleda) stranicu.

Nakon toga je samo potrebno spremiti namještenu varijablu u session scope (natjerati PHP da ju zapamti)

Evo gornje pričice u PHP dijalektu

```
// namještanje session varijable

if (session_is_registered("posjeta")){

$posjeta++;

} else {

$posjeta=1;

}

// registriranje session varijable

session_register("posjeta");
```

Provjeru da li postoji varijabla u koju se sprema broj posjećenih stranica vršimo pomoću funkcije

```
session_is_registered("posjeta")
```

Funkciji se daje string koji mora biti identičan imenu varijable koju provjeravamo. Funkcija vraća true ukoliko varijabla postoji i false u suprotnom.

Nakon te provjere, ovisno o rezultatu se namješta nova vrijednost te varijable.
Pridruživanje se vrši tako da se varijabli koja ima identično ime kao i provjerena varijabla u if-u red prije.

Nakon namještan je potrebno 'registrirati' novu vrijednost varijable pomoću funkcije

```
session_register("posjeta");
```

Ovoj funkciji se također daje string vrijednost koja sadrži identično ime kao i maloprije nesmještena varijabla.

Zašto su ove operacije objavljenje prije switcha koji implementira višenamjensku stranicu?

Zato što je ovo brojenje zajedničko svim stranicama, tj. situacijama. Brojimo ukupan broj otvaranja stranica sitea, a ne samo određene situacije.

Iza switcha u kojem se nalazi sadržaj sitea se ispisuje broj posjećenih stranica.

```
echo "<p>Do sada ste posjetili $posjeta stranica na ovom siteu</p>";
```

Zadatak:

1.

Napisati skriptu koja će pitati korisnika da se učlani na naš site (tražiti nick, lozinku i e-mail) zatim te informaciej spremiti u session varijable te u posebnom odjeljku stranice ispivati ukoliko se korisnik još nije registrirao poruku da se registrira sa linkom na formu, a ukoliko se registrirao umjesto linka ispisati njegov nick koji je odabrao. Treba brojati broj posjećenih stranica. Za skriptu napraviti višenamjensku stranicu na kojoj će se nalaziti sav sadržaj. Neka stranica sadrži uz default za početak stranice te formu za registarciju još barem 2 različita slučaja

Globalne varijable

PHP je opremljen nizom globalnih varijabli u kojima se nalaze razni podatci vezani uz ležeći server, operacijski sustav, informacije o korisniku, cookije te varijable iz formi i Query string.

Popis globalnih varijabli:

- $_POST ili $HTTP_POST_VARS – globalna varijabla sa vrijednostima koje su pristigle iz forme
- $_GET ili $HTTP_GET_VARS – globalna varijabla sa vrijednostima iz query stringa ili komandne linije
- $_SERVER ili $HTTP_SERVER_VARS – globalna varijabla sa informacijama o serveru i korisniku
- $_COOKIE ili $HTTP_COOKIE_VARS – globalna varijabla sa sadržajem cookija
- $_FILES ili $HTTP_POST_FILES – globalna varijabla sa fileovima koji su uplkodani na server putem forme
- $_SESSION ili $HTTP_SESSION_VARS – globalna varijabla sa session varijablama

Globalne varijable su same po sebi nizovi iindeksirani na taj način da se na tekstualnom indeksu jedne nalazi vrijednost te varijable.

Recimo na konkretnom primjeru. Uzmimo da smo negdje na siteu kliknuli na link koji nas odvodi na slijedeću adresu:

http://www.server.com/skripta.php?ime=Kreso&id=123

U ovoj adresi je sve prije znaka ? najobičniji URL koji odvodi korisnika na stranicu. Informacije iza znaka ? spadaju pod tzv komandnu liniju ili Query string. Nadalje u ovom tekstu će se ovo zvati GET metoda.

Nakon što se otvori stranica skripta.php na njoj imamo pristup dvijema varijablama koje su proslijeđene skripti GET metodom. One su spremljene u $_GET globalnu varijblu i možemo im pristupiti na više načina.

Prvi, koji je sigurniji i radit će u svim situacijama je pozivanjem varijable iz globalne varijable $_GET

```
<?

echo $_GET['ime']; // ili echo $HTTP_GET_VARS['ime']

?>
```

GET možemo proslijediti skripti na više načina. Prvi je gore objašnjen – direktnim upisom varijabli u sam URL iza znaka ?.

Drugi način je preko forme koja ima namješten method='GET'. O drugoj metodi će biti više govora u slijedećem odjeljku kada će biti govora o formama.

Generalno pravilo kod globalnih varijabli je da se njihova vrijednost ne može promijeniti direktnim upisom vrijednosti na neki indeks. One su konstante koje namješta sam PHP. Za njihovo namještanje postoje posebne funkcije, ali ovo može biti vrlo velika sigurnosna rupa u aplikaciji. U pravilu globalne varijable namještamo tako da na jednoj stranici namjestimo varijable ili putem forme ili linka sa GET metodom ili registriranjem session ili cookie varijabli. O zadnje dvije metode malo kasnije.

Drugi način za pristup globalnim varijablama je izostavljanjem imena globalne varijable. U ovom slučaju im pristupamo kao i normalnim varijablama preko njihovog originalnog imena.

Za ovo mora u php.ini-ju biti namještena direktiva register_globals na true ili Yes

Za gornji primjer bi to izgledalo ovako:

```
<?

Echo $id;

?>
```

Ova metoda je vrlo nesigurna. Naime, ne znamo kojom metodom podatak dolazi. Ovako neki militantni hacker može poslati varijable iz forme preko GET metode. Ovo je velika sigurnosna rupa i uvijek treba izbjegavati ovakav pristup globalnim varijablama.

PHP I MySQL

Uvod u MySQL

Jedna od najvećih prednosti PHP-a kao serverskog skriptnog jezika je ta što je moguće na vrlo jednostavan način koristiti velik broj bazi podataka. Od onih Microsoftovih preko ODBC-a do naprednih baza poput DB2 i sličnih. Ovaj vodić će proći kroz rad sa MySql-om zato što je ona najpopularnija baza koja se koristi u kombinaciji sa PHP-om. Popularnost ove kombinacije se može pripisati open source prirodi oba „programa" te postojanjem besplatnih verzija za sve glavne operacijske sustave, što olakšava njihovo učenje i korištenje kako početnicima a tako i profesionalcima na ovom području.

Iako su osnove i dizajniranje relacijskih baza podataka sastavni dio područja kojim će se baviti ovaj vodić one neće biti detaljno pokrivene kroz njega. Ovaj vodić će se koncentrirati na upotrebu MySql-a za izradu dinamičkih siteova pomoću PHP-a. Kroz vodić ćete se upoznati sa osnovnim operacijama koje se tiću rada sa MySql-om teoretski i na konkretnom primjeru. Također ćete se upoznati sa osnovama SQL jezika pomoću kojeg se manipulira sa podacima u bazi podataka te nekim od postojećih alata za olakšan rad sa bazama kroz jednostavno grafičko sučelje.

Idemo se nakon ovog kratkog uvoda upoznati sa MySql-om.

MySql

Prije svega idemo nekako pokušati definirati što je MySql. MySql je poslužitelj baza podataka (database server). Drugim riječima, radi se o softwareu kojem se može pristupiti preko mreže na sličan način kao i web (HTTP) poslužiteljima, sa tom razlikom da se MySql-u obično pristupa pomoću korisničkom imena i lozinke.

Na serveru može postojati veći broj baza podataka koje su potpuno samostalne, no unutar jednog projekta se može baratati podacima iz više baza na serveru. Svakom korisničkom računu na serveru je moguće dodijeliti razna administracijska prava na cijeli server ili pojedine baze. Neka od prava bi bila stvaranje novih baza, pravo pristupa postojećim bazama, pravo uređivanja (unosa ili izmejna podataka) postojećih baza itd. Pri instalaciji MySql-a se stvara tzv. superadministartor (obično se zove root) koji ima sva administracijska prava. U vodiću se nećemo previše zadržavati na administarciji samog servera, proći ćemo samo osnovne stvari poput dodavanja korisnika i izmjene njihovih lozinki.

Kao što je već bilo govora u uvodu, jedna od velikih prednosti MySql-a je što postoje verzije za sve važnije operacijske sustave, te ih se distribuira pod GPL licencom (čitaj : besplatno za kučnu upotrebu) što ga čini idealim alatom za učenje osova relacijskih baza podataka te izradu manjih i srednjih siteova čime ćemo se i mi poigrati kroz ovaj vodić. Ovaj vodić je pisan uz predpostavku da je vaš operacijski sustav neka NT verzija Windows OS-a, te sve upute vrijede za tu situaciju. Ukoliko pokušavate koristiti MySql na nekom drugom operacijskom sustavu morate ćete pronaći pomoć na www.mysql.com u dokumentacijama.

Prije nego se upustimo u sočne detalje svih mogućnosti MySql bacimo sa na osnove rada sa MySql-om te spajanja na server i manipuliranjem podataka u tablicama pomoću PHP-a.

Osnove rada s MySqlom

U ovom poglavlju ćemo proći kroz osnove izrada tablica te kroz osnovne tipove polja koje MySql podržava. Da bi olakšali ovaj proces upoznat ćemo se sa alatom za rad sa MySql-om koji je napisan u PHP-u, što ga čini idelanim za naše potrebe. Nakon što kreiramo vrlo jednostavnu bazu spojit ćemo se na MySql server te vidjeti kako PHP-om napuniti bazu podacime, pribavljati podatke iz baze te mijenjati i brisati ih.

Normalno, prije nego li kreiramo bazu moramo otići u nabavku osnovnih namirnica. Prva stvar koja će nam trebati je MySql server koji se može skinuti na www.mysql.com. U ovom trenutku neću opisivati kako instalirati MySql pošto postoje odlične upute u manualu koji se također može naći na www.mysql.com te je poželjno imati ga u nekom windows help file formatu radi lakšeg korištenja kasnije.

Nakon što uspješno instalirate MySql automatski je stvoren ugrađeni korisnik root koji je ujedno i superadministrator. To znači da on ima sve permisije i ima apsolutnu kontrolu i uvid u cijeli server sa svim njegovim bazama. On također može dodavati nove korisnike. Normalno, pošto se radi o ugrađenom korisniku on nema lozinke, i to ćemo sada promijeniti da bi se uskladili sa realnim uvijetima koji se nalaze na komercijalnim serverima.

Namiještanje lozinke korisnika

Lozinku postojećem korisniku možemo namjestiti na više načina. Najjednostavniji je pomoću administarcijskog alata koji dolazi u paketu binarnih fileova u distribuciji. Nažalost, radi se o command prompt alatu tako da ćemo sada morati malo tipkati.

1.Otvorite command prompt (Start -> Run -> Upišite „cmd")

2.Pozicionirajte se u c:/mysql/bin

3. Utipkajte :

```
C:\mysql\bin>mysqladmin.exe –u root password lozinka
```

Ukoliko nekada kasnije želite ponovno izmjeniti lozinku za ovog korisnika ponovite iste korake, samo ovog puta morate mysqladmin-u dati i opciju –p te ćete prije izmjene passworda biti zatraženi stari password.

```
C:\mysql\bin>mysqladmin.exe –u root –p password nova_lozinka
```

Napominjem, ovo je samo jedan od više načina kako izmjeniti ili namijestiti loziku korisniku. Ukoliko se želite upoznati sa ostalim načinima, oni su navedeni i opisani na http://www.mysql.com/doc/en/Passwords.html.

Sada kada smo obavili početne konfiguracije možemo početi koristiti MySql. Prvi alat sa kojim ćemo se ukratko upoznati je defaultni MySql klijent.
Defaultni MySql klijent

MySql u standardnoj distribuciji dolazi i sa klijentom pomoću kojeg možemo pregledavati i manipulirati bazama na našem lokalnom serveru. Radi se o shell programu, koji je poprilično nepraktičan za neki ozbiljniji rad pošto se radi o ASCII prikazima podataka što nerijetko rezultira vrlo nepreglednim rezultatima upita, involvira puno tipkanja i za njegovo korištenje je potrebno solidno poznavanje SQL-a i za najjednostavnije operacije. Iz tih razloga se na njemu neću previše zadržavati, već ću samo proletiti kroz njega i iskoristiti ga za isticanje nekih bitnih stvari.

U tipičnoj instalaciji, klijent se nalazi u c:/mysql/bin direktoriju, a radi se o mysql.exe datoteci. Da bi se klijent pokrenuo potrebno mu je pri pozivanju proslijediti korisničko ime te dati opciju koja omogućuje naknadni upis lozinke.

```
C:\mysql\bin>mysql -u root -p
```

Nakon upisa lozinke ušli ste u mysql klijent koji izgleda odprilike ovako :

```
Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 16 to server version: 4.0.17-max-nt

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

mysql>
```

Kao što se vidi iz uvodnog teksta, dok se nalazite u klijentu svaka naredba mora završiti sa „;“ ili „\g“, dok se jedna naredba može protezati kroz više redova.

Već se iz priloženog može vidjeti da se radi o vrlo neprektičnom alatu u kojem je potrebno svaku pojedinu informaciju, kao popis svih bazi na serveru, zasebno zatražiti. Tako da bi vidjeli sve baze koje postoje na našem lokalnom serveru potrebno je upisati naredbu :

```
mysql>SHOW DATABASES;
```

Prije nego krenemo dalje, važno je napomenuti da su naredbe case insensitive tako da se ne morate brinuti o tome.

Rezultat gornje naredbe bi trebao izgledati otprilike ovako :

```
+---------------+

| Database |

+---------------+

| mysql |

| test |

+---------------+

2 rows in set (0.01 sec)
```

Kao što vidite, pri instalaciji MySQl-a automatski se stvaraju dvije tablice. Ona prva „mysql“ sadrži podatke oko našeg servera poput liste korisnika, njihovih prava i slične informacije. Vrlo je bitno da ne mijenjate podatke unutar te tablice a da niste 100% digurni što radite jer bi moglo trajno oštetiti MySql.

Baza „test“ je prazna i sa njom se možete igrati. Trenutačno ju nećemo dirati, već ćemo napraviti novu bazu te tablicu u njoj.

Ono što je zanimljivo primjetiti je da se ispod rezultata ovog upita ispisuje i vrijeme potrebno za obavljanje upita. Ovo je vrlo zgodan podatak za testiranje i lickanje upita prije njihovog ugrađivanja u samu skriptu. Normalno, ovo nije krajnja brzina izvođenja pojedinog upita unutar same skripte

pošto nešto vremena ode i na prijenos podataka sa Mysql servera do naše skripte, no o ovome će biti govora nešto kasnije.

Vježba 1 : Stvaranje baze kroz MySql shell klijent

Napravit ćemo bazu koja će se koristiti na manjem siteu. Radi se o siteu za malu tvrtku kojoj je potrebno samo periodički objavljivati novosti, dok će ostatak sitea biti statički HTML. Da bi ispunili zahtijeve moramo napraviti novu bazu na serveru u kojoj će se nalaziti jedna tablica.

Naredba za stvaranje nove baze te njen rezultat izgleda:

```
mysql> CREATE DATABASE site_novosti;

Query OK, 1 row affected (0.00 sec)
```

Ovdje je vrlo bitno reći da su imena bazi, tablica te polja u njima case sensitive, te da se u njima ne smije nalaziti bilo kakav prazan prostor (razmak, tabulator, novi red).

Ukoliko sada obavite upit za prikaz svih baza na serveru u listi rezultata će se pojaviti i naša novostvorena baza.Da bi ju mogli početi koristiti te stvarati tablice u njoj ili baratati sa podacima u njima moramo prvo reći MySql-u da ćemo od sada nju koristiti.

```
mysql> USE site_novosti;

Database changed
```

Nakon obavljanja ovog upita / naredbe svi SQL upiti koje upisujemo u komandnu liniju će se odnositi na izabranu bazu.

Slijedeći korak je stvaranje tablice.
Stvaranje tablica

Nakon što smo uspješno stvorili bazu za naš prvi mali projekt stvorit ćemo i tablicu koja će nam biti potrebna da zadovoljimo potrebe našeg sitea. Kao što sam već rekao, jedini dinamički koji je potreban je vrlo jednostavan sistem novosti.

Struktura tablice će izgledati ovako :

U gornjoj shemi oznaka PK označava Primary Key ili Primarni ključ. Ukratko, primarni ključ je ono polje tablice koje će za svaki redak tablice imati unikatnu vrijednost te tako nepogrešivo identificira svaki pojedini redak tablice. U situacijama kada iz samog entiteta koji reprezentira tablica, kao u našem slučaju, nije vidljiv neki segment koji bi unikatno identificirao svaki zapis, polju se daje cijelobrojni (integer) tip, po mogućnosti što većeg raspona. U MySql-u postoji mehanizam koji može automatski dodjeljivati novu vrijednost takvom polju pri svakom unosu novog retka u tablicu. Takva polja se zovu autoincrement. Suprotna situacija bi bila kada iz samog objekta (ili entiteta, onoga što tablica reprezentira) vidimo neki njegov segment koji unikatno identificira svaki zapis u tablici. Primjer toga bi bio JMBG kod tablice koja bi sadržavala listu klijenata, ili neka posebna šifra sa vlastitom logikom generiranja. Mi ćemo koristiti autoincrement cjelobrojno polje.

U našoj tablici upoznati ćemo se sa još tri druga tipa polja.

Naslov će biti varchar tipa, ograničen na maksimalno 150 znakova. Tekst će biti tipa text, što je tip koji omogućuje upis velikih binarnih podataka (u ovom slučaju teksta) i nema ograničene veličine a datum, pogađate, tipa date.

Za sada neću ulaziti u detalje ovih i ostalih MySql tipova podataka.

Naredba za stvaranje tablice izgleda ovako :

```
CREATE TABLE novosti (

idnovost INT( 11 ) NOT NULL AUTO_INCREMENT ,
naslov VARCHAR( 150 ) NOT NULL ,
tekst TEXT NOT NULL ,
datum DATE NOT NULL ,
PRIMARY KEY ( idnovost )

);
```

Nakon uspješnog izvršenja ove naredbe možemo provjeriti da li je ona stavrno stvorena izvrašavanjem slijedeće naredbe :

```
mysql> SHOW TABLES;

+----------------------------+

| Tables_in_site_novosti |

+----------------------------+

| novosti |

+----------------------------+

1 row in set (0.01 sec)
```

Na samoj naredbi za stvaranje tablica se neću zadržavati jer ćete kasnije u vodiču biti upoznati sa alatom za rad sa MySql-om kroz grafičko korisničko sučelje kroz koje je puno brže i lakše stvoriti novu tablicu.

Posjetite za detaljnije informacije: http://dev.mysql.com/doc/mysql/en/creating-tables.html

Da bi izašli iz shell klijenta upišiti :

```
mysql>quit;
```

Ovime ćemo i završiti ovaj dio vodića i prebacit ćemo se na zanimljivi dio. Spajanje MySql-a i PHP za stvaranje dinamičkih web stranica.

PHP i MySQL

Evo nas napokon na zanimljivom dijelu ovog vodiča. Kroz slijedećih pra primjera ćemo napraviti HTML interface na bazu koju smo stvorili u dosadašnjim primjerima. Uz same primjere će biti diskretno opisana i jedna od metoda strukturiranja i planiranj aplikacija. Radit će se o vrlo jednostvnoj shemi jer se radi o vrlo jednostavnoj aplikaciji, no vi koji želite zagristi u kompleksnije projkete kasnije u vodiću će biti primjer kako isplanirati i strukturirati veće aplikacije. Za sada ćemo se držati osnova i jednostavnih primjera.
Spajanje na MySql server

Spajanje na MySql server je prva operacija koju moramo obaviti u skriptama koje ovise o podacima u bazi. Spajanje se mora uspješno obaviti tako da pri obavljanju svakog od koraka spajanja moramo provjeravati da li je operaciaj uspješno obavljena.

Kao što to obično biva sa serverima, oni se nalaze na mreži i njima se pristupa pomoću IP-a ili domene (hosta). Radi današnjih sigurnosnih trendova u većini situacija se MySql server i HTTP server nalaze na istom računalu, tj. na localhostu, te se MySql-u obično i pristupa na taj način. Uz podatak o hostu servera su nam još potrebni korisničko ime i lozinka da bi se uspješno spojili na server te komunicirali sa njim. Ukoliko koristite neki zakupljeni prostor te informacije će vam dostaviti hosting provider pri stvarajnu vašeg računa. Za naš lokalni rad ćemo koristiti korisnika root te lozinku koju smo za njega stvorili na početku ovog vodića.

Prije nego se bacimo na sam primjer stvorit ćemo direktorij na našem lokalnom HTTP serveru gdje ćemo spremati sve naše primjere te kroz njih napraviti cijelu aplikaciju koju smo počeli raditi stvaranjem prve baze i tablice u njoj.

Radi mogućih razlika u direktorijskim putevima između različitih srevrera (Apache, IIS idrugi) ja neću navoditi točne puteve do tih direktorija, već ću se truditi pisati sve adrese relativno na web root direktorij servera (u slučaju Apacheja npr „C:/apache/htdocs/" ili IIS-a „C:/inetpub/wwwroot/"). Ukoliko će se negdje u primjerima stvoriti potreba na navođenjem potpune lokalne adrese (full local path) to će biti posebno nataknuto.

Također, svi primjeri će biti pisani i obrađeni na taj način da vam neće biti potreban nikakv code editor već ćete primjere moći pisati i u svom omiljenom editoru. Ovdje bi se želio zadržati na mitu notepada koa alata za kodiranje. Iako ćete na internetu na mnogo mjetsta naći „gurue" koji govore da je notepad sve što vam treba ja se ne bi složio sa tom tvrdnjom. Pri kodiranju većih siteova nerijetko se nađete u situaciji da ispred sebe imate skriptu od 1000+ lijija koda, a PHP vam javlja MySql error u liniji 478. Pošto notepad nema ispi broja linije očito ej da ćete dosta vremena izgubiti tražeći pravu liniju. No uz tu osobinu dobrog code editora notepadu fale mnoge druge stvari koje ima svaki osrednji i bolji IDE (Integrated Developmnet Enviroment) poput rada na projektnoj bazi umjesto na bazi pojedinog dokumenta / skripte itd. Ovo govorima samo zato da se ne bi osječali manje vrijednima ili, ne daj bože, glupima kada čitate na usenetu da neko svoje PHP skripte piše u notepadu. Tu se samo radi o čovjeku koji voli sam sebi otežavati život a nije garancija njegovog znanja. To rečeno, idemo dalje.

U svojem web root folderu web servera stvorite direktorij imena „php_mysql". U njega ćemo spremati sve dokumente iz slijedećih par primjera.

Primjer 1 : Spajanje na MySql server

U ovom primjeru ćemo napraviti file koji ćemo nazvati „dbspoj.php" te ga pohraniti u direktori koji smo upravo stvorili. U njemu će se nalaziti kod koji će se spojiti na MySql server. Ovaj file ćemo koristiti u kasnijim primjerima kada ćemo raditi konkretne operacije nad bazom.

Otvorite svoj omiljeni code editor i utipkajte slijedeći kod.

```
<?

if (!$db=@mysql_connect("localhost", "root", "lozinka"))

{

die ("<b>Spajanje na mysql server je bilo neuspješno</b>");

}

if (!mysql_select_db("site_novosti", $db))
```

```
{

die ("<b>Greška pri odabiru baze</b>");

}

?>
```

Ukoliko ste MySql funckijama dali točne podatke rezultat skripte bi trebala biti prazna stranica.

Objašnjenje :

Idemo pogledati sve ključne dijelove ove skripte te što i kako svaki od njih radi.

Samo spajanje na server se obavlja pomoću mysql_connect funckije. Idemo se na trenutak na njoj zadržati. Funkciji se proslijeđuju tri argumenta. Prvi je host na kojem se nalazi server. Kao što sam već prije rekao, ovdje može doći IP ili hostname servera, no skoro uvijek se radi o alliasu localhost na IP 127.0.0.1 (barem kod većine komercijalnih hosting providera). Drugi argument je korisničko ime sa kojm se pokušavamo spojiti, a treći lozinka za tog korisnika.

Ukoliko su svi dani podaci točni, te je mysql pristupačan i operacionalan funckija će vratiti „spoj" na bazu. Ovdje sam pojednostavio što funckija točno vraća, no ukratko se radi o vrijednsoti pomoću koje PHP kasnije može obavljatu komunikaciju sa serverom te ju je potrebno pohraniti u varijablu. Ukoliko je neka od informacija netočna ili se nešto desilo sa samim MySql serverom funckija vraća false.

Kao što vidite, kraj samog poziva mysql_connect funkcije se nalazi znak @. On služi za ignoriranje grešaka tj. onemogučuje PHP da ispisuje greške koje se dogode u naredbi iza njegovog pojavljivanja. U našem slučaju bi PHP ispisao grešku u kojoj je sadržano između ostaloga i korisničko ime sa kojim se pokušavamo spojiti na server, što je potencijalni sigurnosni rizik pa ne želimo cijelom svijetu obznaniti tu informaciju. Ukoliko vas zanima što se događa ukoliko njega ne bi tamo bilo izmejnite lozinku za korisnika u nešto što nije točno i maknite @. Normalno, pri izradi i testiranju skripte je nerijetko potrebno dobiti što detaljniji opis greške, pa je za to vrijeme @ nepotreban, no dobra praksa se zaštiti prije nego skriptu date cijelom svijetu na korištenje.

Još jedna vrlo bita stvar je provjeravati uspjeh operacije spajanja. Naime, radi se o tome da će sav ostali kod koji barata sa podaciam u bazi očekivati da smo se uspješno spojili na serevr. Ukoliko se ne spojimo taj kod će se ipak obavljati, normalno neuspješno, te će se vaša lijepa, brižno dizaunirana stranica pretvoriti u rugobu unakaženu PHP errorima. Da ne kažem da se opet izlažemo sigurnosnim rizicima jer će se pri ispisu tih grešaka opet prikazivati informacije koje su potencijalni sigurnosni rizici. Stoga je najbolje odmah pri spajanju provjeriti da li je operacija uspješno obavljena, te ukoliko nije prekinuti izvršavanje skripte pomoću die funckije sa ispisom greške koja nam govori u kojem koraku je stvar zapela.

Inače, pri izradi skripti, kod savijesnih programera, dobar dio kodova uvijek otpada na prihavćanje i obradu potencijalnih grešaka, od kojih se neke rijetko ili nikada neće dogoditi, tj. mora doći do ekstremnih uvijeta da se jave. No takvi kodovi imaju dozu kvalitete i sigurnosti koji se nepostojani kod brzopletih i ljenih programera te izazivaju frustraciju kod korisnika. Kao što vidite, u našem primjeru bi bilo dovoljno 2 reda da se obave sve operacije, no mi smo ih izveli u 4 (ako se izostavi formatiranje koda), što je duplo više.

Slijedeći korak pri spajanju na MySql server je odabir baze na njemu. Ako se sječate nešeg rada sa shell klijentom tamo smo također morali prvo odabrati bazu u kojoj smo stvorili tablicu. Tako i u našim skriptama moramo odabrati bazu sa čijim ćemo tablicama baratati kroz skriptu pomoću mysql_select_db funckije. Mysql_select_db funckija kao prvi argument dobija ime baze koju želimo koristiti, a kao drugi vezu tj spoj na MySql server gdje se ta baza nalazi. Pošto smo spoj (rezultat mysql_connect funckije) pohranili u varijablu $db nju proslijeđujemo na drugo mijesto.

Želio bih se na trenutak zadržati na tom drugom argumentu. On nije obavezni argument, tj. mogli smo ga izostaviti te mysql_select_db funckiju pozvati u mysql_select_db("site_novosti") obliku i sve bi i dalje radilo. Radi se o tome da PHP automatski pamti posljednju otvorenu vezu na MySql server te ju uvijek po defaultu koristi u svim mysql funckijama gdje je taj argument izostavljen (recimo u funkciji koja izvršava neki SQL upit na serveru). No, ukoliko se u našoj skripti spajamo na dva različita servera, ili ostvaramo dvije zasebne veze na isti server i ne koristimo taj argument u MySql funckijama PHP će uvijek koristiti onu vezu koja je zadnja otvorena. Takoda u toj situaciji moramo proslijeđivati onu varijablu koja u sebi sadrži onaj spoj na kojem želimo ivršiti neku operaciju. Praktični primejr ovakve situacije bi bio kopirannje baze na loklanom serevru u bazu na nekom remote serveru (iako za ovu operaciju postoje bolji, gotovi alati koje ćemo kasnije upoznati).

Uspjeh operacije odabira baze također moramo provjeriti jer ukoliko je tu nastala greška skripta će se nastaviti obavljati, ali kodovi koji komuniciraju sa MySql serverom će javljati greške, tako da je potrebno prekinuti rad skripte ako je nastala greška.

Sada kada smo se uspješno spojili na MySql server idemo vidjeti koja sve čuda možemo raditi sa MySql-om. Točni slijed slijedećih koraka pri izradi naše aplikacije nije unaprijed određen, no da bi slijedili neku logiku i sebi malo olakšlai sma proces razvijanja aplikacije krenut ćemo sa formom za upis podataka u bazu po logici da prvo moramo imati podatke u bazi da bi ih mogli pregledavati i uređivati.

Unos podataka

Ovo će biti prvi kompliciraniji primjer koji ćemo napraviti pošto se proces unosa podataka u bazi sastoji od dva osnovna koraka. Prvi je prikupljanje podataka kroz formu, a drugi pohrana tih podataka u bazi. Kasnije ćemo proširiti ovaj model sa dodatnim koracima validacije upisanih podataka te će sam proces postati cikličan a ne ravan kao sada. Naime, u slijedećem primejru će jednostavnosti izrade radi korisnik morati kliknuti Back gumb na svom browseru ukoliko se želi vratiti na formu, dok ćemo kasnije napraviti kod koji će mu omogućiti ispunjavanje forme sve dok uspješno ne pohrani podatke u bazu. No o „filozofskoj" strani ovoga u svoje vrijeme.

Prije nego krenemo na sam primjer idemo pogledati kako pomoću SQL-a upisujemo podatke u bazu. Za unos podataka u SQL-u postoji INSERT naredba. MySql ima dvije sintakse za nju, no mi ćemo koristiti onu verziju koja je kompaltibilna i sa ostalim bazama poput Postgresql-a, Microsoft Jet-a (Access) i MsSql Servera. Na taj način ćemo kasnije uvijek moći prebaciti (portati) naše skripte i na druge baze ukoliko se za to pojavi potreba bez većih izmjena kodova.

Ukoliko vas zanima i druga sintaksa te detalji obje pogledajte
http://dev.mysql.com/doc/mysql/en/INSERT.html.

Idemo pogledati INSERT naredbu za naš konkretan primjer pa ćemo proći kroz njene detalje.

```
INSERT INTO novosti (naslov, tekst, datum) VALUES ("Prva novost", "Ovo je tekst prve novosti koji moze biti puno veci od ovoga", "2004-04-12");
```

Objašnjenje :

Sve SQL naredbe počinju nazivom tipa naredbe. Pošto trenutačno obavljamo insert naredbu navodimo njeno ime. Prije navođenja imena tablice u koju želimo pohraniti podatke dolazi INTO. Kod naziva tablice je vrlo bitno napisati njeno ime u pravilom „caseu" jer su imena baza, tablica i polja case sensitive. Sada dolazimo do kompliciranog dijela. Kao što vidite, u zagradama su navedena polja naše tablice, no jedno fali. Polje idnovost nije navedeno i ne radi se o pogrešci u pisanju. Naime, ukoliko se neko polje izostavi u listi njemu se automatski upisuje defaulta vrijednost (koja se može postaviti pri stvaranju tablice). Mi smo pri stvaranju tablice polju idnovost dali auto_increment osobinu koja pri svakom insertu tom polju pridružuje po defaultu vrijednost veću za

1 od zadnje upisane, stoga da bi se ta defaultna vrijednost sama upisala, polje sa auto_increment osobinom moramo izostaviti iz ove liste.

Nakon liste polja dolazi VALUES čime naznačujemo da slijedi lsita vrijednosti koja se pohranjuje u tablicu. Ovdje dolazi jedna jako bitna stvar na koju treba paziti pri pisanju upita. Vrijednost koju želimo pohraniti u jedno od polja mora biti na istoj poziciji u listi vrijednosti kao i ime polja u koje ju želimo spremiti u listi polja. Znači u našem primjeru se vrijednost „Prva novost" pohranjuje u polje naslov itd.

Kod stvaranja liste polja nije bitan redoslijed. Bitno je jedino da se ime polja u svojoj listi pozicijom poklapa sa vrijednosti u listi vrijednosti.

Siguran sam da ste primjetili čudan način pisanja datuma. 2004-04-12 nije uobičajen način pisanja datuma i na prvi pogled nije jasno što je mjesec a što je dan u njemu. Format slovima bi bio GGGG-MM-DD. MySql pohranjuje datume u ovom formatu radi jednostavnost sortiranja datuma pošto su sve vrijednosti složene po svom utjecaju na poziciju u sortiranoj listi tako da se mogu soritirati jednostavnim i brzim algoritmima kao i tekstualni podaci. Kod unosa datuma u bazu morate uvijek paziti da je datum pravilnog oblika, što će reći da 2004-1-10 neće proći jer mu fali druga znamenka za mjesec. Isto vrijedi i za dan u mjesecu. Znaći prepravljeno bi izgledalo 2004-01-10. No, ovo nije veći problem iz dva razloga. Prvi je taj što MySql ima ugrađene funckije koje vraćaju trenutno vrijeme na serveru, tako da se može nju koristiti za upis trenutnog datuma, a druga je što sam PHP ima date() funckiju koja ima mogućnost fleksibilnog generiranja datuma, tako da možemo bez muke generirati datum i ovog oblika.

Idemo se sada prebaciti na konkretan primjer koji će upotrijebiti gornji SQL upit da pohrani podatke forme u bazu. Za sada ćemo se držati osnovnih stvari, tako da će validacije forme i sve napredne mogućnosti biti preskočene radi bolje razumljivosti primjera.

Primjer 2 : Forma za unos podataka

Ovaj primjer je nastavak izrade sitea iz prvog primjera te ćemo u njemu koristiti dbspoj.php koji smo napravili u prvom primjeru. Otvorite svoj omiljeni code editor i u njemu stvorite unosvijesti.php te ga pohranite u isti direktorij gdje se nalazi i dbspoj.php.

```
<html>

<head>

<title>Unos vijesti</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h1>Unos novosti</h1>

<hr>

<?

if (!$_POST["SBunos"])

{

?>
```

```
<form method="post" action="">

Naslov : <input type="text" name="naslov"><br>

Tekst :&amp;nbsp;&amp;nbsp; <textarea name="tekst"></textarea><br>

<input type="submit" name="SBunos" value="Pohrani">

</form>

<?

} else {

include "dbspoj.php";

$datum=date("Y-m-d");

$sql="INSERT INTO novosti (naslov, tekst, datum)

VALUES ('$_POST[naslov]', '$_POST[tekst]', '$datum')";

if (mysql_query($sql))

{

echo "Novost je uspješno pohranjena";

} else {

echo "Nastala je greška pri pohrani novosti<br>" . mysql_error();

}

}

?>

</body>

</html>
```

Objašnjenje :

Na ovom primjeru ćemo naučiti par vrlo bitnih stvari. Prva nije direktno vezana uz PHP ili MySql, no može biti vrlo bitna i problematična. Radi se o kodnoj stranici (encodingu) našeg dokumenta.

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

Da ne bi nailazili na probleme pri radu sa hrvtaskim znakovima nije toliko bitno koji ćete encoding odabrati, već da koristite uvijek isti kroz sve dokumente projketa. Pravilo bi bilo da će hrvatski znak koji je u bazu pohranjen pod jednom kodnom stranicom pravilno biti prikazan samo pod tom istom kodnom stranicom.

Struktura, tj. logika ove skripte slijedi logiku same operacije. U njoj je prvi korak unos podataka u formu, a drugi korak je pohrana tih podataka u bazu. Kao uvijet obavljanja prvog ili drugog koraka je postojanje vrijednosti u varijabli $_POST["SBunos"], tj. varijable koju stvara pritisnuti submit gumb forme imena SBunos.

Idemo malo pogledati samu formu.

```
<form method="post" action="">

Naslov : <input type="text" name="naslov"><br>

Tekst :   <textarea name="tekst"></textarea><br>

<input type="submit" name="SBunos" value="Pohrani">

</form>
```

Kao što vidite, u form tagu action atribut nema vrijednost. Ovo će uzrokovati slanje podataka istoj stranici na kojoj se nalazi forma. Znači, u našem primjeru se forma prikazuje na unosvijesti.php te će se i nakon submitanja forme opet otvarati ista stranica. Također, ukoliko je pri otvaranju stranice postojao neki query string iza imena stranice i on će se također koristiti pri otvaranju nove stranice nakon submitanja forme. Ovo je vrlo korisno jer kasnije uvijek možemo promijeniti ime stranice / dokumenta iz unosvijesti.php u nešto drugo,a skripta će i dalej raditi bez ikakvih izmejna u njoj ili dati dodatan uvijet prikaza forme na osnovi neke vrijednosti u nekoj query string varijabli.

Prije nego uđem u detalje drugog koraka želim napomenuti da ovaj primjer nije primjeren za neki relani projekt jer mu fale validacije podataka forme, no taj dio je trenuto izostavljen radi jednostavnosti primjera te bolje razumljivosti ključnih stvari koje on ilustrira.

Drugi korak pohranjuje podatke iz forme u bazu. Da bi se moglo nešto pohraniti u bazu prvo se moramo spojiti na MySql server i odabrati ju, što smo napravili u prvom primjeru koji je pohranjen u dbspoj.php te ga ovdje samo includamo.

```
include "dbspoj.php";
```

Nakon uspješnog spajanja na bazu se možemo zabaviti sa samim generiranjem SQL upita.

```
$datum=date("Y-m-d");
```

U našem primjeru moramo u bazu upisati datum kada je novost stvorena, tj. trenutačni datum. Ovo je moguće napraviti na više načina. Prvi je ovaj koji je korišten u primjeru pomoću PHP date() funckije, adrugi bi bio pomoću ekvivalentne MySql ugrađene funkcije. Koju ćete koristiti je amnje bitno, no moj prijedlog je da se koristi PHP funkcije pošto se pomoću nje može stvoriti bilo koji prošli ili budući datum dodavanjem drugog opcisjkog argumenta (za detalje pogledati opis date() funkcije u PHP manualu) što je vrlo korisno kada nam se server nalazi u drugoj vremenskoj zoni te ga moramo uskladiti sa našim lokalnim vremenom. Kao što je već bilo govora, datum se u MySql upisuje u točno predodređenom GGGG-MM-DD formatu, što je zadano date funckiji njenim argumentom kojim određujemo format datuma. Siguran sam da pogađate da se Y zamijenjuje sa 4 znamenke trenutne godine, m sa trenutnim mjesecom u dvije znamenke te d sa danom u trenutnom mjesecu u dvije znamenke. Za popis svih mogućih znakova koje će date funckije obraditi tj. izmjeniti sa nekim segmentom datuma pogledajte ops date() funckije u PHP manulau. Mi ćemo krenuti dalje.

```
$sql="INSERT INTO novosti (naslov, tekst, datum)

VALUES ('$_POST[naslov]', '$_POST[tekst]', '$datum')";
```

Slijedeći korak je stvaranje stringa u kojem se nalazi sam SQl upit za unos podataka u tablicu. Pošto smo njegove detalje već upoznali prije ovog primjera samo ću napomenuti par stvari. Prva stvar je da pri ovakvom generiranju upita trebate jako paziti na navodnike, tj da se ne dogodi situacija da neki otvorite a ne zatvorite ili da neke od njih potpuno izostavite, poput onih ' u gornjem retku koji služe za navođenje tekstualnih (string) podataka unutar samog upita. Oko ovoga će vam djelomično pomoći kvalitetan code editor koji boja stringove posebnom bojom te time naglašava gdje koji string počinje a gdje završava ili ima neku drugu metodu naglašavanja istoga. Druga bitna stvar je da pri radu sa formama uvijek koristiti ime globalne vrijable ($_POST, $_GET itd.) iz koje podaci dolaze radi sigurnosti da netko ne pokuša kroz query string na neki način prevariti našu skriptu. No ovo je izvan domene ovog vodića tako da se na pojašnjivanju ovoga trenutno neću zadržavati.

```
if (mysql_query($sql))

{

echo "Novost je uspješno pohranjena";

} else {

echo "Nastala je greška pri pohrani novosti<br>" . mysql_error();

}
```

Ovo je glavni dio naše skripte koji obavlja samu komunikaciju sa MySql serverom, tj. izvrašav SQL upit na njemu. To se obavlja pomoću mysql_query() funckije. Idemo se malo na njoj zadržati.

mysql_query() funckija prima dva argumenta. Prvi je string podatak koji sadrži valjani upit (SQL ili neku MySql naredbu) koji je obavezan, dok je drugi neobavezan, a radi se o otvorenom i valjanom spoju na MySql server koji smo mi u našj dbspoj.php skripti pohranili u $db varijablu. Pošto mi u našim primjerima baratamo samo sa jednim spojem on je izostavljen i PHP će ga automatski koristiti kao što je već objašnjeno u prvom primjeru.

U slučaju INSERT upita mysql_query() vraća true ukoliko je upit uspješno obavljen ili false u suprotom. Iz tog razloga je moguće pomoću if-a provjeriti da li je naša novost pohranjena u bazu te ovisno o tome ispusujemo odgovarajuću poruku korisniku. Moguće greške koje se tu obično znaju javljati su ili sintaktičke priprode, tj. nešto je krivo napisano u samom SQL upitu ili tehničke prirode kao recimo da se nešto dogodilo sa MySql serverom ili vezom na njega iza našeg uspješnog spajanja. Također je moguće da je korisnik u formu upisao neki nedozvoljen znak poput ' a mi ga našim validacijskim procesima nismo uhvatili i obradili što na kraju rezultira opet sintaktičkom pogreškom u samom SQL upitu.

U slučaju da je nastala neka od ovih čestih ili neka druga rijeđa greška možemo ispisati njen opis pomoću mysql_error() funckije. Ona može primiti jedan argument sa nekim otvorenim i valjanim spojem na MySql server. Bitno je da bude isti onaj nad kojim je i obavljen upit. Pošto se mi spajamo samo na jedan serevr, ovaj argument je opet izostavljen.

Prije nego nastavmo, unesite par vijesti u bazu pomoću naše forme tako da stvorimo kup podataka sa kojim ćemo moći baratati u kasnijim primjerima.

I to bi bila pohrana podatak u bazu. Slijedeći korak u izradi naše aplikacije je izrada alata pomoću kojeg ćemo moći prolistati sve novosti koje se nalaze u našoj tablici. Ovo ćemo kroz par slijedećih primjera proširivati da bi na kraju dobili kompleksan alat koji će biti centar administracije cijelog sisteam novosti iz kojeg će se moći brisati postojeće novosti, te prebacivati na formu za uređivanje pojedine novosti. No ići ćemo redom. Prvo ćemo izabrati sve novosti iz baze te ih ispisati u tablici.

Pribavljanje podataka

Prije nego se uhvatimo samog PHP dijela priće idemo vidjeti osnove SQL upita za pribavljanje podataka. Pošto se radi o najkompleksnijoj vrsti upita koju ćemo koristiti u našim skriptama njega ćemo postepeno upoznavati kroz slijedeće primjere na teoretskoj i praktičnoj osnovi.

Najjednostavniji od njih je upit koji pribavlja sve retke iz jedne tablice.

```
SELECT * FROM novosti
```

Idemo prokomentirati strukturu gornjeg upita. Upit počinje nazivom vrste upita. Ovo je pravilo koje dijele svi upiti za manipulaciju podacima u osnovnim oblicima (neki upiti imaju proširene oblike koji služe za obavljanje nekih kompleksnijih operacija ili obavljaju analizu samog upita, no mi se time nećemo baviti u ovom vodiću). Nakon imena operacije slijedi lista polja koja će se pojaviti u tablici rezultata upita. Sada ste sigurno zbunjeni, jer u našoj tablici ne postoji polje *. Zvjezdica je skraćenica koja govori da želimo pribaviti sva polja iz navedenih tablica.

Nakon liste polja slijedi FROM kojigovori da slijedi lista tablica iz kojih se podaci pribavljaju. Mi ćemo se za sada zadržati samo na pribavljaju podataka iz jedne tablice, a iz više tablica ćemo pribavljati malo kasnije u vodiću zbog dodatnih komplikacija u logici i sintaksi spajanja dvije ili više tablica u jedan rezultat upita. No bez brige, imao što do onda raditi tako da nam neće biti dosadno do tada.

Ovo su ujedno i svi obavezni dijelovi svakog SQL SELECT upita te ukoliko jedan od njih izostavite MySql će javiti grešku i neće obaviti upit.

Otvorite shell mysql klijent i obavite opisani upit. Nakon drugog primjera njegov rezultat bi mogao izgledato odprilike ovako :

```
mysql> SELECT * FROM novosti;

+----------+--------+-------+--------+

| idnovost | naslov | tekst | datum |

+----------+--- ----+-------+--------+

| 1 | Prva novost | Ovo je tekst prve novosti koji moze ... | 2004-04-12 |

| 2 | Druga novost | Ovo je tekst druge novosti.

ima par redaka, no ipak nije prevelika. | 2004-04-14 |

| 3 | Cibona nije pobijedila | Cibona nije uspijela pobijediti u finalu goodyear
lige na razočaranje svih nas | 2004-04-18 |

| 4 | Cisterna se okrenula | Danas se okrenula cisterna i nastala je... | 2004-04-19
|

+----------+--- ----+-------+--------+

4 rows in set (0.01 sec)
```

Kao što vidite, rezultat je vrlo nepregledan, i to je razlog zbog kojeg ćemo kasnije u vodiću upoznati bolji alat za manipuliranje podacima u bazi, no za sada je shell klijent ima bolju edukacijsku svrhu pošto morate ručno unostiti sve naredbe te preporučam da izdržite još malo sa njim. U gornjem ispisu su neke vijesti naknadno skraćene radi bolje preglednosti ispisa.

Rezultat našeg upita nije sortiran po niti jednom polju, iako se to tako može činiti iz gornjih podataka. No retci su vraćenu po redoslijedu upisa u bazu, pa je logično da su sortirani i po datumu unosa od manjeg prema većem te po idnovost polju pošto se radi o auto_increment polju koje svakom novom retku daje vrijednost veću za 1 od onog prošlog.

Idemo sada vidjeti kako bi sortirali podatke na onaj način kojim mi želimo. Reci, po datumu, samo od najvećeg datuma prema najmanjem.

```
SELECT * FROM novosti ORDER BY datum DESC
```

Da bi sortirali rezultate upita dodajemo mu ORDER BY klazulu u ovom slučaju na njegov kraj. Oko točnog redosliejeda pojedinih klazula unutar upita se trenutno nemojte brinuti. Na kraju ovog dijal vodića se nalazi točan redoslijed svih klazula koje ćemo sada upoznati.

ORDER BY mora biti popraćen imenima polja po kojima želimo sortirati rezultat te način sortiranja. U našem slučaju se radi o polju datum i DESC (descending) načinom, tj. silazno. Suprtotno, tj. uzlazno smo mogli sirtirati tako da izostavimo DESC ili umjesto njega upišemo ASC (ascending). Naime, ukoliko se izostavi smjer sortiranja rezultata defaultni smjer je uzlazan.

Ukoliko rezultat želimo sortirati po više polja, recimo po naslovu novosti i datumu unosa možemo koristiti slijedeći upit:

```
SELECT * FROM novosti ORDER BY naslov, datum DESC

mysql> SELECT * FROM novosti ORDER BY naslov, datum DESC;

+----------+--- ----+-------+--------+

| idnovost | naslov | tekst | datum |

+----------+--- ----+-------+--------+

| 3 | Cibona nije pobijedila | Cibona nije uspijela pobijediti u finalu goody... |
2004-04-18 |

| 4 | Cisterna se okrenula | Danas se okrenula cisterna i nastala je prava... |
2004-04-19 |

| 2 | Druga novost | Ovo je tekst druge novosti.

ima par redaka, no ipak nije prevelika. | 2004-04-14 |

| 1 | Prva novost | Ovo je tekst prve novosti koji moze biti puno ve... | 2004-04-12
|

+----------+--- ----+-------+--------+

4 rows in set (0.01 sec)
```

Kao što vidite iz rezultata, rezultat je sortiran tako da se prvo sortira po naslovu, a tek onda po datumu. Znaći da će drugo polje po kojem želimo sortirati biti uzeto u obzir samo u slučaju da dva ili više redaka dijele istu vrijednost u prvom polju po kojem sortiramo. Ovo vrijedi i za sva ostala polja u listi polja po kojim asortiramo, tj. utjecaj na sortiranje rezultata nekog polja ovisi o njegovoj poziciji unutar liste polja za sortiranje rezultata upita.

Za sada ćemo se zaustaviti na ovim klazulama SELECt upita, a ostale naprednije ćemo upoznati kroz primjere koji slijede iza ovoga.

Naš slijedeći korak je izrada centralnog administracijskog alata sistema novosti (ovo je zvučalo jako važno). Zapravo će se raditi o stranici u kojoj će se nalaziti lista svih novosti iz tablice sa popratnim alatma poput linkova za brisanje pojedine novosti, linkom za uređivanje pojedine novosti itd.

Primjer 3 : Preglednik novosti

Kao što zaključujete, ovo je još jedna komponenta našeg sitea. Tako da otvorite svoj omiljeni code editor te u istom direktoriju kao i prijašnja dva filea (dbspoj.php i unosvijesti.php) stvorite novi imena preglednik.php. Ovaj alat ćemo graditi kroz par slijedećih primjera, tako da ćemo početi sa osnovnim kosturom te ga polako nadograđivati

```
<html>

<head>

<title>Preglednik novosti</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h1>Administracija novosti</h1>

<hr>

<a href="unosvijesti.php">Dodaj novost</a>

<hr>

<?

/*

Funkcija koja pretvara datum iz MySql oblika u obični format

*/

function pretvoriDatum($mysqlDatum)

{

$tmp=explode("-", $mysqlDatum);

$datum=$tmp[2] . "." . $tmp[1] . "." . $tmp[0];

return $datum;

}

include "dbspoj.php";

$sql="SELECT idnovost, naslov, datum FROM novosti ORDER BY datum DESC";
```

```
if (!$q=mysql_query($sql))

{

echo "Nastala je greška pri izvođenju upita<br>" . mysql_query();

die();

}

if (mysql_num_rows($q)==0)

{

echo "Nema novosti";

} else {

?>

<table width="760" border="0" cellpadding="1" cellspacing="1">

<tr>

<td><b>Naslov</b></td>

<td><b>Datum</b></td>

<td><b>Opcije</b></td>

</tr>

<?

while ($redak=mysql_fetch_array($q))

{

?>

<tr>

<td><?=$redak["naslov"]?></td>

<td><?=pretvoriDatum($redak["datum"])?></td>

<td>

<a href="uredivanjenovosti.php? idnovost=<?=$redak["idnovost"]?>">Uredi</a>

<a href="?action=obrisi&idnovost=<?=$redak["idnovost"]?> ">Obriši</a>

</td>

</tr>
```

```
<?

}

?>

</table>

<?

}

?>

</body>

</html>
```

-

Objašnjenje :

Evo nas na malo kompleksnijem primjeru. SELECT upiti su najopširniji. Razlog tome je prilično očit, jer uz obavljanje upita moramo te podatke i pribaviti tj. u ovom slučaju ispistai u obliku tablice. No idemo reddom.

```
function pretvoriDatum($mysqlDatum)

{

$tmp=explode("-", $mysqlDatum);

$datum=$tmp[2] . "." . $tmp[1] . "." . $tmp[0];

return $datum;

}
```

Prva stavr koja se nalazi u dokumentu je funckija za obradu datuma. Kao što je vć bilo govora u vodiću, MySql pohranjuej datume u obliku GGGG-MM-DD dok nama treba DD.MM.GGGG. radi bolje čitljivosti i razumljivosti. Ova funckija pretvara MySql datum u datum kojeg smo navikli gledati. Radi se o prilično jednostavnoj operaciji.

```
$tmp=explode("-", $mysqlDatum);
```

Prvo u neku privremenu varijablu pomoću explode funkcije razbijemo datum u njegove sastavne dijelove. Za detalje explode funckije pogledajte PHP manual, no ukoliko ste lijeni to sada raditi objasnit ću ukratko što ona radi. Ona kao ulazne podatke dobija dva stringa. Prvi je znak ili string koji služi kao uvijet razdijele drugpog ulaznog argumenta. Drugim rijećima, string koji se proslijeduje kao drugi argument funkcije biva „razbijen" na više dijelova na taj način da prvi element rezultatnog niza sadrži znakove od početka tog stringa do prvog pojavljivanja znaka ili stringa iz prvog argumenta, drugi element sadrži znakove između prvog i drugog pojavljivanja znaka ili stringa iz prvog argumenta, dok zadnji element rezultata sadrži znakove od zadnjeg pojavljivanja sznaka ili stringa iz prvog argumenta funkcije pa do kraja stringa koji želimo razbiti. No mislim da ovo objašnjenje uopće nije bilo potrebno jer je svejasno iz slijedeće linije koda koja koristi taj razbijeni datum tj. niz stvoren iz njega za stvaranje novog stringa.

```
return $datum;
```

Ukoliko niste upoznati sa detaljima ponašanja i izrade funckija u PHP-u, return vraća danu mu vrijednsot u glavni program tako da se ta vrijednost može kasnije koristiti u skripti. Ovo će biti puno jasnije kada vidite kako smo iskoristili funkciju u našem primjeru.

Prvi kod same skripte je spajanje na MySql server te odabir baze, za što opet koristimo dbspoj.php koji smo napravili u prvom primjeru.

```
include "dbspoj.php";
```

I evo nas na zabavnom dijelu primjera. Prva stvar koju moramo napraviti je stvoriti upit.

```
$sql="SELECT idnovost, naslov, datum FROM novosti ORDER BY datum DESC";
```

Pošto su detalji ovog upita već objašnjenji prije ovog primjera na njemu se neću zadržavati. Reći ću samo da nije obavezno upite upisivati u varijablu te onda tu varijablu proslijeđivati mysql_query() funckiji. Mogli se sam upit napisati ravno u mysql_query() funckiju. Npr. ovako:

```
mysql_query("SELECT idnovost, naslov, datum FROM novosti ORDER BY datum DESC")
```

No, moje osobno mišljenje je da pohranom upita dobijamo čišći kod, te kasnije uvijek možemo opet koristiti taj upit recimo pri ispisu nekog error reporta gdje prvo ispišemo upit u kojem je nastala greška te nakon toga ispišemo grešku pomoću mysql_error() funkcije te na taj način dobijemo jasniju sliku što je uzrokovalo grešku u upitu. Ovo dobija na težini kada dođemo do primjera koji će dinamički generirati sam SQL gdje mogu nastatai sintaktičke greške zbog nepravilno upisanih podataka na osonovi kojih se generira sam upit, recimo upisivanje navodnika u polje za pretraživanje sitea.

```
if (!$q=mysql_query($sql))

{

echo "Nastala je greška pri izvođenju upita<br>" . mysql_query();

die();

}
```

Kao što vidite, ovog puta je kod koji izvršava SQL upit na serveru nešto drugačiji nego kod drgog primjera. Ovog puta smo pohranili rezultat mysql_query funkcije u varijablu. Ovo je obavezno jedino kod izvršavanja SELECT upita dok kod izvršavanja ostalih tu pohranu možemo komotno izostaviti da se operaciaj uspješno obavi. Naime, da bi nakon izvršenja upita mogli pribaviti podatke koje taj upit vraća moramo sačuvati „pokazivač" na njegove rezultate tj. detalje. Ovdje se ne bih previše zadržavao na tehničkim detaljima koji to sve uvjetuju, dovoljno je zapamtiti da je potrebno zapamtiti taj pokazivač (u PHP manualu se zove result identifier) ukoliko želimo prihvatiti bilo kakav povratni podataka sa MySql servera nakon izvršenja upita. Iako se kod neSELECT upita može izostaviti da bi se upit izvršio ipak je u nekim situacijama potrebno zapamtiti result identifier. Recimo, nakon UPDATE upita se može provjeriti koliko je redaka u tablici izmjenjeno, i da bi prihvatili taj podatak moramo sačuvati result identifier.

U slučaju SELECT upita link identifier koristimo za provjere vezane uz upit te za pribavljanje skupa rezultata koji je upit pribavio.

Sigurno se pitate koje provjere možemo i trebamo obaviti u vezi samog SELECT upita a da se ne radi o pribavljanju rezultata upita. Radi se o slijedećoj problematici. Mysql_query funckija vraća false samo u slučaju da je nastala neka greška pri obavljanju samog upita. Znaći neka sintaktička

pogreška ili nešto tehničke prirode što je obično izvan sfere PHP-a. No mysql_query neće vratiti false ukoliko upit nije vratio niti jedan rezultat u slučaju da uvijeti upita nisu zadovoljeni ili je tablica iz koje pribavljamo podatke prazna. U cilju izrade skripte koja neće ispisivati neke PHP / MySQL errore u toj situaciji moramo prije ispisa samog rezultata upita provjeriti da li ima rezultata za prikaz te ukoliko nema poduzeti odgovarajuće korake da se korisnik obavijesti o tome te mu po potrebi ponuditi opcije što može učiniti u toj situaciji.

Tu provjeru obavljamo sa :

```
if (mysql_num_rows($q)==0)

{

echo "Nema novosti";

} else {

...

}
```

Magična funckija koja obavlja samu provjeru je mysql_num_rows() koja kao argument prima result identifier te vraća integer broj koji je jednak ili veći od nule. Normalno, result identifier mora biti valjan, tj. ova funckija će javiti error ukoliko je pri obavljanju upita nastala greška tj. mysql_query() je vratio false vrijednost umjesto result identifiera. Znaći moramo paziti da se ova funckija ne poziva ukoliko skripta nije prošla obavljanje upita. To smo u našoj skripti osigurali die() funckijom ukoliko je nastala greška pri izvršavanju upita.

Logika našeg primjera je poprilično jednostavna. Ukoliko upit nije pribavio niti jedan rezultat korisniku se javlja poruka o tome, a ukoliko je skripta nastavlja sa ispisom rezultata u obliku tablice (zamijenjeno sa ... u gornjem isječku).

I evo nas na glavnom dijelu naše skripte za kojeg smo se pripremali u svim njenim dosadašnjim koracima, a to je ispis samih rezultata upita. Rezultate ispisujemo u tablici radi bolje preglednosti i razumljivosti, što je vrlo bitno kod izrade dinamičkih siteova sa bazama podataka jer sav naš trud i znanje PHP-a i relacijskih baza podataka pada u vodu ukoliko korisniku nije jasno što gleda i kako se to koristi čim to prvi put vidi. No ovo ispada iz sfere ovog vodiča, no ipak je građa za malo samostalnog konstruktivnog razmišljanja.

Idemo malo analzirati ispis podataka.

```
?>

<table width="760" border="0" cellpadding="1" cellspacing="1">

<tr>

<td><b>Naslov</b></td>

<td><b>Datum</b></td>

<td><b>Opcije</b></td>

</tr>

<?
```

Kao što vidite, prije samg ispisa smo izašli iz PHP moda u HTML te ispisali prvi redak tablice sa nazivima stupaca. Prvi stupac smo rezervirali za nalsov novosti, drugi za datum njenog upisa a treći za opcije. Opcije su ništa drugo nego linkovi koji pokreću nove operacije nad vijesti koja se nalazi u istom retku tablice kao i opcije. O ovome par redaka kasnije.

Nakon što smo ispisali „headere" tablice vraćamo se u PHP mode i krećemo sa ispisom podataka.

```
<?

while ($redak=mysql_fetch_array($q))

{

?>

<tr>

<td><?=$redak["naslov"]?></td>

<td><?=pretvoriDatum($redak["datum"])?></td>

<td>

<a href="uredivanjenovosti.php? idnovost=<?=$redak["idnovost"]?>">Uredi</a>

<a href="?action=obrisi&idnovost=<?=$redak["idnovost"]?> ">Obriši</a>

</td>

</tr>

<?

}

?>

</table>

<?

...
```

Prije nego krenem sa pojašnjem samog koda potrebno je shvatiti kako se rezultati preuzimaju sa MySQL servera. Rezultate se može prihvatiti samo jedan po jedan i to samo unaprijed. Znači nije moguće krenuti od zadnjeg rezultata iz skupa već od prvog prema zadnjem. Ukoliko ih se želi prihvatiti obrnutim redoslijedom to se može učiniti izmjenom samog SQL upita ili pohranom svih rezultata u niz pa naknadnim sortiranjem tog niza.

Svaki pojedini redak možemo prihvatiti sa jednom od više mogućih funckija. U našem primjeru se koristi mysql_fethc_array(), a na raspolaganju su nam još mysql_fetch_row(), mysql_fetch_assoc(), mysql_fetch_object(). Sve funckije se pribavljaju cijeli redak rezultata, samo se razlikuju u obliku u kojem ga vraćaju.

Mysql_fetch_array() vraća redak ako niz koji se sastojih od svih polja rezultata s tim da je svako polje indeksirano brojem i svojim imenom. Da ovo ilustriram na našem primjeru. Mi pribavljamo

polja : idnovost, naslov, datum. To će reći da će naš niz koji vrati mysql_fetch_array() imati šest elemenata. Za svako polje po dva. Na indeksu 0 i idnovost će se nalaziti vrijednsot koja je pohranjena u stupcu idnovost unutar retka koji smo upravo pribavili pozivanjem funckije nad našim skupom skupom rezultata, na indeksi 1 i naslov će se očito nalaziti vrijednost koaj se nalazi u polju naslov unutar tog retka rezultata te na indeksu 2 i datum će se nalaziti vrijenost stupca datum. Ovdje je vrlo bitno napomenuti da numeričke vrijednosti ovise o poziciji polja u listi polaj za pribavaljanje u samom SQL upitu, tako da ukoliko izmjenimo tu listi izmejnit će se i raspored polja u nizu koji vrati mysql_fetch_array() funckija. No tekstualno indeksiranim poljima nije bitan raspored već samo imena koja smo naveli u toj listi polaj za pribavljanje što ovu metodu čini puno fleksibilnijom za korištenje te ćete taj način naći u svim primjerima u ovom vodiču, ali i u većini skripti koje možete naći na internetu.

Mysql_fetch_row() također vraća niz, sa tom razlikom da u njemu postoje samo numerički indeksi.

Mysql_fetch_assoc() isto vraća niz, no u njemu se nalaze samo tekstualno indeksirana polja.

Znaći, od sve tri navedene funckije najkompletnija je prva, mysql_fetch_array() te je moj prijedlog da ju koristite. U nekim vodićima ćete naići na teze da je najbolje koristiti mysql_fetch_row funckiju jer je najbrža, no čak i da je to istina razlika u brzini bi se mjerila i milisekundama, a imali bi teže shvatljiv kod koji bi kasnije teže i održavati. Brzina skripti koje koriste bazu podataka ponajviše ovise o količini upita koje skripta obavlja te o kompleksnosti samih upita. Što je upit kompleksniji to su sporiji. Tu recimo spadaju upiti koji pribavljaju podatke iz dvije ili više tablica sa kojima ćemo se kasnije upoznati, te upiti sa kompleksnim uvijetima pribavljanja i pravilima sortiranja. Stoga se nemojte zabrinjavati ukoliko se naviknete koristiti mysql_fetch_array funckiju i onda pročitate da je ona sporija.

Četvrta navedena funckija, mysql_fethc_object() umjesto niza vraća objekt stvoren od polja retka, no pošto objekti nisu spominajni nigdje dalej u ovom vodiću neću se zadržavati na ovoj funckiji. Ukoliko se želite bolje upoznati sa njom, ali i sa drugim MySql funckijama otvorite si MySql Functions dio PHP manuala.

Sve gore navedene funckiej imaju zajedničko to da će vratiti false na kraju našeg skupa rezultata. Ovo čini while petlju idealnim alataom za pribavljanje svih rezultata jer će se obaviti onoliko puta koliko ima rezultata u skupu. Još jedna zajednička stvar svim tim funckijama je da kao svoj argument dobijaju result identifier koji vraća mysql_query() funckija.

```
while ($redak=mysql_fetch_array($q))
```

U bloku naredbi while petlje ispisujemo jedan redak tablice te u pojedini stupac tog retka ispisujemo pribavljenje vrijednosti poput <?=$redak["naslov"]?> s čime ispisujemo naslov novosti.

Ovdje bi se želio zadržati na trenutak na dijelu opcija vezanih uz svaku novost.

```
<a href="uredivanjenovosti.php? idnovost=<?=$redak["idnovost"]?>">Uredi</a>

<a href="?action=obrisi&idnovost=<?=$redak["idnovost"]?> ">Obriši</a>
```

Korisniku nudimo dvije moguće opcije. Brisanje i uređivanje pojednine novosti. Kao što vidite, u linku smo naveli kao href stranicu koju još nismo napravili, no uz malo planiranja možemo predvidjeti njeno ime i ponašanje. Pošto se još nismo upoznali sa detaljima upita za brisanje i izmjenu podataka za sada neću komplicirati. Ono što je bitno uočiti je da nam je za uspješno obavljanje tih operaciaj potreban idnovosti pomoću kojeg ćemo kasnije moći identificirati koju novost želimo obrisati ili obraditi. Također, vidite da ćemo brisanje novosti obaviti unutar iste ove skripte koju smo upravo napravili, samo što će ju biti potrebno otvoriti sa predodređenim query stringom koji će naznačiti da želimo obrisati novost.

I to bi bilo to, nakon ispisa svih redaka rezultata moramo samo poaziti da pozatvaramo sve HTML tagove i sve vitičaste zagrade da bi skripta bila bez grešaka i gotovi smo. Ništa lakše.

Sada ćemo nastaviti sa DELETE upitom za brisanje podataka, a SELECT upitu sa njegovim ostalim mogućnostima ćemo se vratiti na kraju ovog dijela osnova SQL-a pošto nam u ovom trenutku nisu neophodne.

Brisanje podataka

Kao i do sada, prije nego li se bacimo na PHP primjer izvedne brisanja podataka prvo ćemo proći kroz osnove SQL upita kojim brišemo podatke.

```
DELETE FROM tablica
```

Kao što vidite, sam upit je poprilično jednostavan i lako ga se pamti. U prijevodu gornja naredba znači „Obriši sve iz tablice imena tablica". No ovo je očito situacija koja će se rijetko događati u nekim realnim skriptama, već se obično radi o brisaju jednog ili manje grupe podataka, te je to brisanje obično uzrokovano direktnom naredbom samog korisnika kroz grafičko sučelje.

U našoj skripti koju gradimo kroz zadnjih par primjera imamo potrebu za brisanjem jedne novosti na osonovi njezinog ID-a. Idemo pogledati kako bi izgledao upit koji obavlja tu operaciju.

```
DELETE FROM novosti WHERE idnovost=1
```

Gornji upit će obrisati samo onu novost koja ima idnovost jednak 1. Kao što primjećujete, upoznali smo novu klazulu SQL jezika. Mislim na WHERE klazulu. Where služi za naznačivanje da nakon njega slijede uvijeti pomoću kojih se ograničava skup podataka nad kojima će se obaviti neka operacija. U ovom slučaju smo odabrali skup od jedne novosti koju želimo obrisati. Where se može koristiti i u ostalim vrstama SQL upita, a najčešće se koristi u SELECT upitima, no o upotrebi where-a u njima će biti puno više govora kasnije u ovom poglavlju. Za sada ćemo biti jendostavni te ćemo samo malo preraditi gornji primjer tako da briše sve novosti koje su između dva datuma.

```
DELETE FROM novosti WHERE datum>='2004-03-01' AND datum <'2004-04-01'
```

U prijevodu, zadali smo MySql-u da obriše sve novosti iz trećeg mjeseca. Kao što vidite, unutar WHERE klazule je moguće kombinirati više uvijeta koji ograničavaju skup podataka nad kojima se obavlja operacija kombiniranjem logičkim operatorima. Praktički se WHERE može shvatiti kao if u PHP-u koji se obavlja nad svakim retkom tablice. Normalno, nemojte me tu doslovce shvatiti jer se ne radi o tome, ali ilustracije radi se može tako shvatiti.

Također, pomoću DELETE naredbe nije moguće obrisati pojedinu vrijednost unutar nekog polja tablice. Recimo, nije moguće obrisati naslov novosti čiji je idnovst jednak 1. Za to se koristi UPDATE naredba koju ćemo upoznati odmah nakon slijedećeg primjera.

Prije nego krenemo na sam primjer želio bih nešto istaknuti. Pošto mi trenutno radimo vrlo jednostavnu skriptu koja koristi samo jednu tablicu u bazi operacija brisanja je poprilično jednostavna operacija. No ukoliko bi se naš projekt sačinjavao od više, međusobno povezanih tablica morali bi paziti da kada brišemo podatke iz jedne tablice obrišemo ili na naki drugi način ukinemo vezu sa podacima koji su u relaciji sa obrisanim podatkom u ostalim tablicama baze. Kasnije u vodiču ćemo se upoznati sa nekim mehanizmima koji će se sami brinuti o tim problemima za nas, no nerijetka početnička greška je brisati podatke samo iz jedne tablice, dok se vezani podaci ostavljaju neizmjenjeni, što sa vremenom dovodi do narušavanja integriteta podataka, te time sama relacijska baza tj njena podatkovna struktura gubi svoj smisao. No trenutno zaboravite da sam išta rekao, sa ovom problematikom ćete biti upoznati nešto kasnije.

Primjer 4 : Brisanje novosti

Kao što je već rečeno, brisanje novosti ćemo izvesti u istom dokumentu / skripti iz prošlog primjera. Iz tog razloga je ubačeni kod posebno naglašen u donjoj skripti. Nenaznačeni kod je identičan onome iz prošlog primejra, te ga se u objašnjenju primjera nećemo opet doticati.

```
<html>

<head>

<title>Preglednik novosti</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h1>Administracija novosti</h1>

<hr>

<a href="unosvijesti.php">Dodaj novost</a>

<hr>

<?

/*

Funkcija koja pretvara datum iz MySql oblika u obični format

*/

function pretvoriDatum($mysqlDatum)

{

$tmp=explode("-", $mysqlDatum);

$datum=$tmp[2] . "." . $tmp[1] . "." . $tmp[0];

return $datum;

}

include "dbspoj.php";

// brisanje novosti

if ($_GET["action"]=="obrisi")

{

if ($_GET["idnovost"])

{
```

```
$sql="DELETE FROM novosti WHERE idnovost=" . $_GET["idnovost"];

if (mysql_query($sql))

{

echo "Novost je uspješno obrisana";

} else {

echo "Nastala je greška pri brisanju novosti<br>" . mysql_error();

}

}

}

$sql="SELECT idnovost, naslov, datum FROM novosti ORDER BY datum DESC";

if (!$q=mysql_query($sql))

{

echo "Nastala je greška pri izvođenju upita<br>" . mysql_query();

die();

}

if (mysql_num_rows($q)==0)

{

echo "Nema novosti";

} else {

?>

<table width="760" border="0" cellpadding="1" cellspacing="1">

<tr>

<td><b>Naslov</b></td>

<td><b>Datum</b></td>

<td><b>Opcije</b></td>

</tr>

<?

while ($redak=mysql_fetch_array($q))
```

```
{

?>

<tr>

<td><?=$redak["naslov"]?></td>

<td><?=pretvoriDatum($redak["datum"])?></td>

<td>

<a href="uredivanjenovosti.php? idnovost=<?=$redak["idnovost"]?>">Uredi</a>

<a href="?action=obrisi&idnovost=<?=$redak["idnovost"]?> ">Obriši</a>

</td>

</tr>

<?

}

?>

</table>

<?

}

?>

</body>

</html>
```

-

Objašnjenje :

Idemo pogledati dio koji smo dodali postojećoj skripti :

```
// brisanje novosti

if ($_GET["action"]=="obrisi")

{

if ($_GET["idnovost"])

{
```

```
$sql="DELETE FROM novosti WHERE idnovost=" . $_GET["idnovost"];

if (mysql_query($sql))

{

echo "Novost je uspješno obrisana";

} else {

echo "Nastala je greška pri brisanju novosti<br>" . mysql_error();

}

}

}
```

Kao što vidite, prije nego obavljamo samo brisanje moramo provjeriti da li su ispunjeni svi uvijeti za obavljanje brisanja. U našem slučaju smo si rekli da moraju postojati dvije varijable u query stringu da bi se pokrenulo brisanje novosti. Prvo je varijabla koja opsije koju operaciju želimo obaviti. Mislim na $_GET["action"] varijablu. Nju smo komotno mogli izostaviti pošto je jedina dodatna opercija koju obavljamo u ovoj skripti uz pribavljanje i izlistavanje novosti brisanje novosti. No, radi eventualnog proširivanja ove skripte da može obavljati dodatne operacije, te radi dodatne razumljivosti samog koda ubacili smo taj dodatni uvijet pošto nas ništa ne košta napisati tu dodatnu liiniju koda.

Nakon ustanovljavanja da se radi o pokretanju brisanja moramo provjeriti da li imamo ID novosti koju želimo obrisati pošto njega ubacujemo ravno u SQL naredbu za brisanje, te ukoliko ona fali greška pri obavljanju upita je neizbježna. U našoj skripti je to učinjeno vrlo jednostavnom provjerom postojanja neke vrijednosti različite od 0 u varijabli $_GET["idnovost"]. Tu bi se želio malo zadržati i napomenuti da ovo nije niti približno dovoljna provjera. Siguran sam da pogađate i zašto. Vrlo je očito da se id novosti koja se šalje na brisanje može ručno upisati u adresu tj query string kojim otvaramo skriptu te na taj način obrisati novost kojoj pogodimo ID. No, iz istog razloga se u query string može upisati i neka tekstualna vrijednost koja ne može biti validan Id novosti, što će sigurno izazvati grešku pri obavljanju upita. Iz tog razloga bi tu trebalo provjeriti da li je vrijendost u $_GET["idnovost"] numerička, te po mogućnosti veća od nule pošto su naši ID-evi svi veći od nule.

No, sva ova priča spada u temu sigurnosne teorije, tako da više neću ići u njenu dubinu radi koncentriraanj na stvari koje su nama trenutno bitne, no prije nego se bacite na izradu velikih siteova sa potencijalnom širom publikom otvorenog / public tipa predlažem da dobro promislite kako se osigurati od militantnih korisnika koji će pokušati na bilo koji način „srušiti" vaše sigurnosne postavke. No, mi idemo dalje.

Nakon provjera ispunjenja svih uvijeta brisanja krećemo na samo brisanje.

```
$sql="DELETE FROM novosti WHERE idnovost=" . $_GET["idnovost"];

if (mysql_query($sql))

{

echo "Novost je uspješno obrisana";

} else {
```

```
echo "Nastala je greška pri brisanju novosti<br>" . mysql_error();

}
```

Detalje SQL upita smo obradili prije primjera tako da ćemo ih sada preskočiti. Ono što bih samo želio istaknuti je da smo u ovom primjeru po prvi put dinamički generirali sam SQL upit sa dinamičkim podacima. Iako ovo zvuči vrlo high-tech radi se o čistom ljepljenju varijable u SQL upit, no kasnije u vodiču će biti puno kompleksnijih primjera, tako da ste sada vidjeli što znaći napraviti dinamički upit.

Samo izvršavanje upita je vrlo slično kao kod INSERT upita. Dovoljno je obaviti upit pomoću mysql_query() funckije. Čak nije obavezno pohraniti njen rezultat u neku varijablu pošto nemamo potrebe za njenim korištenjem kasnije u skripti.

Očito, ukoliko je mysql_query() vratio true možemo predpostaviti da je novost obrisana a ukoliko je vratio false znaći da je nastala neka greška koju će nam mysql_error() opisati.

Tu sada dolazimo do jedne zanimljive stvari. Pokušajte obrisati jednu vijest, te kada vam se opet otvori stranica sa porukom o uspješnom brisanju vijesti refreshajte stranicu (tako da opet otvorite starnicu sa istim query stringom). Kao što vidite, opet nam se javila ista poruka o uspješnom brisanju novosti, no mi znamo da ta novost više ne postoji u bazi. Što se tu dogodilo? Kao što sma već rekao, mysql_query() samo izvršava upit te vraća true ukoliko je on uspješno obavljen, a ne zna detalje njegovog obavljanja, tj. u ovom slučaju ne zna da li je neki redak stavrno obrisan u tablici.

Da bi provjerili da li je naša novost stvarno obrisana možemo koristito još jednu mysql funkciju koja se nalazi u PHP-u. Radi se o mysql_affected_rows() funckiji. Kao što joj ime kaže, ona vraća broj redaka koji su bili utjecani posljednjom operacijom. Ona se može koristiti samo nakon DELETE, UPDATE i INSERT upita, tj. upita koji rade izmjene nad podacima tablica.

Idemo vidjeti kako bi ju implementirali u našu skriptu.

```
$sql="DELETE FROM novosti WHERE idnovost=" . $_GET["idnovost"];

if (mysql_query($sql))

{

if (mysql_affected_rows() > 0 )

{

echo "Novost je uspješno obrisana";

} else {

echo "Nije obrisana niti jedna novost!";

}

} else {

echo "Nastala je greška pri brisanju novosti<br>" . mysql_error();

}
```

Znaći, dodali smo još jednu provjeru nakon obavljanja upita brisanja koja ispituje da li je broj obrisanih redaka veći od nule. Ako sada pokušate refreshati stranicu nakon brisanja, ili ručno izmjeniti idnovost u query stringu na neki nepostojeći ID novosti prikazat će vam se pravilna poruka.

I to je to. Polako ali sigurno se približavama kraju naše skripte. Ostalo nam je još samo upoznati se kako urediti postojeći podatak čime ćemo zaokružiti osnove SQL-a te time i osnove korištenja MySql-a u PHP skriptama. Nakon toga ćemo se na trenutak vratiti na postojeće gradivo da bi izradili dio sitea koji će vidjeti i naši korisnici, te ćemo malo doraditi postojeće alate da budu dodatno jednostavniji i pregledniji. Tu ćemo već početi uranjati u malo naprednije vode, no u njih ćemo do grla zaroniti u slijedećem poglavlju vodiča.

Uređivanje podataka

Sada moram priznati da sam malo lagao sa gornjim naslovom. Naime, nećemo se odmah poigrati sa samim upitima za uređivanje podataka u tablicama. Zašto? Razlog tome je zapravo poprilično jednostavan. U realnim web aplikacijama, kojima se mi trenutnu bavimo, da bi mogli urediti neki podatak morate ga prvo pribaviti podataka iz baze, prikazati ga u formi te tek nakon izmjena pohranjujeme izmjene na njemu.

Možda primjećujete da smo već prošli kroz osnove pribavljanja podataka. No, isto tako se sigurno sjećate da nismo prošli kroz sve klazule SELECT naredbe već smo rekli da će o njima biti govora kasnije. Pogađate, kasnije nas je sustiglo.

Kada smo pričali o SELECT naredbi upoznali smo se kako prihvatiti sve retke jedne tablice te kako ih sortirati. Sada ćemo vidjeti kako izabrati točno određene retke ili redak tablice. U našem konkretnom slučaju moramo pribaviti jednu novost da bi ju mogli obraditi. Pa idemo vidjeti kako to učiniti.
Izdvajanje skupa podataka tablice

Znaći, cilj nam je izdvojiti samo dio redaka (podataka) tablice. Možda već pogađate kako ćemo to učiniti jer smo to već radili pri brisanju podataka u tablici. Tamo smo koristili WHERE klazulu da bi odabrali novost koju smo htjeli obrisati. Tako ćemo ju i sada koristiti, i to na identičan način pošto se opet radi o izabiranju samo jedne novosti.

```
SELECT * FROM novosti WHERE idnovost=1
```

Gornji primjer pribavlja sva polja novosti čiji je ID jednak jedan, ukoliko takva postoji u tablici. Idemo se malo zadržati na WHERE-u pošto se radi možda o najbitnijoj tj. najkorištenijoj klazuli. Kao što smo već vidjeli pri upoznavanju sa DELETE upitom unutar uvijeta WHERE klezule se može kombinirati veći broj uvijeta spojenih logičkim funckijama I, ILI, ISKLJUČIVO ILI (XOR) te NOT. Sam uvijet je moguće zakomlicirati koliko nam mašta dopušta, no pri takvim upitima morate imati na umu da će s ekompleksnost uvijeta odraziti na brzinu obavljanja upita, što bi moglo značajno usporiti pribavljanje podataka iz vrlo velikih tablica (u tablicama sa X00 000 + redaka, gdje je X = 1 do 9). Taj efekt se može donekle ublažiti korištenjem indeksa, no sa njima ćemo se upoznati kasnije tako da ćemo ovo za sada ostaviti. Umjesto toga idemo vidjeti primjer jednog kompleksnijeg upita nad našom tablicom.

```
SELECT * FROM novosti WHERE (idnovost<10 AND !datum < "2004-01-01") OR datum =
"2004-04-19"
```

Ili u prijevodu, izaberi sva polja redaka iz tablice novosti čiji je idnovost manji od 10 a da im datum nije manji od 1.1.2004. ili one retke čiji je datum jednak 19.4.2004.

Normalno, ovakav upit nema previše smisla, no poslužio je ilustraciji kompleksnosti koja se može naći u uvijetu WHERE klazule.

Još jedna napomena. U gornjem primjeru su zagrade mogle komotno biti izostavljene. To ima veze sa prednosti pri izvršavanju logičkih operacija. Slično kao i u klasičnoj aritmetici, u logičkoj se prvo obavlja AND, tek zatim OR (množenje i zbrajanje), no ovo je tema u koju ne bih ulazio pošto bi vam trebala biti poznata iz osnova programiranja. Zagradama sam se poslužio samo da vas upoznam sa činjenicom da ih se može koristiti.

Ne znam jeste li primjetili da do sada nisma koristio niti jedno od dva tekstualna polja iz naše tablice u primjerima. To sam učinio namjerno jer taj slatkiš čuvam za sam kraj. Sada ćemo nastaviti sa pravim naslovom ovog dijela vodiča, a to je uređivanje podataka u tablici.
Izmjena postojećih podataka

Za izmjenu podataka se koristi UPDATE naredba. Sa njom smo se upoznali sa svim vrstama upita. Idemo se bez puno priće prebaciti na konkretan primjer.

```
UPDATE novosti SET naslov="Novi naslov"
```

Idemo malo ovo razložiti. Kao i svi ostali upiti do sada i ovaj počinje navođenjem imena operacije koju želimo izvesti. No, ovog puta smo odmah nakon naziva naredbe naveli tablicu u kojoj želimo napraviti izmjene. Nakon imena tablice slijede SET koji govori da slijede parovi imena polja tablice sa pridruženim novim vrijednostima. Nakon izvršavanja gornjeg primjera bi izmjenili svim retcima tablice naslov u „Novi naslov".

Ukoliko je potrebno izmjeniti više polja istovremeno, to bi učinili ovakvim upitom:

```
UPDATE novosti SET naslov="Novi naslov", datum="2004-04-21"
```

Znaći, za izvođenej izmejna vrijednosti u više polja potrebno je dva naziva polja sa pridruženim vrijednostima odvojiti zarezom.

Normalno, izmejna vrijednsoti svim retcima neke tablice je vrlo rijetka pojava u realnim skriptama. Iz tog razloga je moguće pomoću WHERE klazule ograničiti skup podataka nad kojima se vrši izmjena. U našem konkretnom primjeru bi to izgledalo odprilike ovako :

```
UPDATE novosti SET naslov="Novi naslov", datum="2004-04-21" WHERE idnovost=1
```

Ovim upitom smo izmjenili naslov i datum upisa novosti koja ima ID jednak jedan.

Kao što vidite, ne radi se o nečemu previše problematičnom. Zato neću više teoretizirati već se bacam na konkretan primjer.

Primjer 5 : Izmjena novosti

Evo nas na najkompleksnijem primjeru do sada jer se u njemu koriste dva različita tipa upita. Prvi koji pribavlja podatke koji se mogu izmjeniti te drugi koji ih mijenja u bazi. Zato nećemo gubiti vrijeme već otvorite svoj omiljeni code editor i upišite slijedeći kod.

```
<html>

<head>

<title>Uređivanje novosti</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

```
</head>

<body>

<h1>Uređivanje novosti</h1>

<hr>

<?

include "dbspoj.php";

if (!$_POST["SBuredi"])

{

// provjera ulaznih podataka

$err=false;

if (!$_GET["idnovost"])

{

echo "Nepotpuni ulazni podaci";

$err=true;

} else {

// pribavi novost za prikaz u formi

$sql="SELECT * FROM novosti WHERE idnovost=" . $_GET["idnovost"];

if (!$q=mysql_query($sql))

{

echo "Nastala je greška pri izvođenju upita<br>" . mysql_query();

$err=true;

} elseif (mysql_num_rows($q)==0) {

echo "Nepostojeća novost";

$err=true;

} else {

$novost=mysql_fetch_array($q);

}

}
```

```
// prikaz forme ukoliko je prihvaćena

if (!$err)

{

?>

<form method="post" action="">

<input type="hidden" name="idnovost" value="<?=$_GET["idnovost"]?>">

Naslov : <input type="text" name="naslov" value="<?=$novost["naslov"]?>"><br>

Tekst : <textarea name="tekst"><?=$novost["tekst"]?></textarea><br>

<input type="submit" name="SBuredi" value="Pohrani">

</form>

<?

}

} else {

// pohrana izmjena novosti

$sql="UPDATE novosti SET naslov='". $_POST["naslov"] ."', tekst='".$_POST["tekst"]
."' WHERE idnovost=". $_POST["idnovost"];

if (mysql_query($sql))

{

if (mysql_affected_rows() > 0 )

{

echo "Novost je uspješno uređena.";

} else {

echo "Novost nije izmjenjena.";

}

} else {

echo "Nastala je greška pri izmjeni novosti<br>" . mysql_error();

}

}
```

```
?>

</body>

</html>
```

-

Objašnjenje :

Kao što vidite, sedamdeset linija koda u PHP-u nije mnogo kada se uzme u obzi da se radi o ponešto jednostavnoj operaciji. Tako da se ne začudite kada vam prvi malo veći projekt pređe broj od par tisuća linija. Ali idemo se baciti na bitne stvari.

Kao i u dosadašnjim primjerima, na samom početku skripte smo includali dbspoj.php. Nadam seda je razlog tome već poznat pa vas neću opet gnjaviti sa tim objašnjenjem već prelazim na prvi kora, a taj je popunjavanje forme sa postojećim podacima iz baze.

Primjećujete varijablu $err? Ona nam služi kao zastavica koja govori da li su podaci uspješno pribavljeni iz baze te da li je sigurno prikazati formu. Kao i svaka zastavica, ova može biti dignuta ($err=true) ili spuštena ($err=false). Ako pogledate ostatak koda koji slijedi vidjet ćete da zastavicu dižemo kada nešto pođe po zlu, te formu ne prikazujemo u slučaju da je zastavica dignuta. Normalno, na početku skripte moramo stvoriti zastavicu u njenom spuštenom stanju iz razloga koje diktira zdrav razum. Na početku skripte nije još ništa pošlo po zlu.

Prije samog pribavljanja podataka iz baze moramo obaviti neke provojere.

```
if (!$_GET["idnovost"])

{

echo "Nepotpuni ulazni podaci";

$err=true;

} else {

...
```

Ovime provjeravamo da li je skripti prolijeđena informacija ID-a novosto koju želimo urediti kroz formu. Očito je da nam je ovo ključna informacija jer na osnovu nje prihvaćamo novost sa kojom ćemo popuniti formu, te ju u slijedećm koraku koristimo za određivanje koju vijest uređujemo unutar UPDATE upita.

Nakon što smo se uvjerili da imamo sve potrebne podatke za obavljanje željenih operacija možemo prihvatiti novost.

```
$sql="SELECT * FROM novosti WHERE idnovost=" . $_GET["idnovost"];

if (!$q=mysql_query($sql))

{

echo "Nastala je greška pri izvođenju upita<br>" . mysql_query();
```

```
$err=true;

} elseif (mysql_num_rows($q)==0) {

echo "Nepostojeća novost";

$err=true;

} else {

$novost=mysql_fetch_array($q);

}
```

Ovaj dio bi već trebao biti poznat, samo što smo ga ovog puta malo drugačije napisali. Umjesto serije if-ova je korištena if ... elseif ... else struktura. Ona nam ovdje više odgovara zbog svoje osobine da se može spuštati po uvijetima sve dok se ne dođe na nekog koji je uzrokovao grešku ili ukoliko nije došlo do greške se obavlja else grana strukture. Ovo zadovoljava naš uvijet da se niže operacije ne smiju obaviti ukoliko je neka od viših operacija završila sa greškom. Znaći, ne možemo izvršiti mysql_num_rows() funckiju ukoliko je mysql_query vratio false, tj. ukoliko postoji greška u upitu ili je nastala neka druga greška na samom serveru.

```
$novost=mysql_fetch_array($q);
```

Sada se sigurno pitate zašto nije korištena while petlja za pribavljanje retka rezultata upita. Razlog je zapravo prilično očit. Naš upit može vratiti samo jedan rezultat kojeg prihvaćamo samo jednim pozivanjem mysql_fetch_array() funckije tako da bi isti rezultata postigli i korištenjem while petlje.

Nakon pribavljanja novosti možemo krenuti na prikaz forme. No prije samog prikaza forme moramo provjeriti našu zastavicu, tj. provjeriti da li je bilo grešaka u dosadašnjim obavljneim operacijama.

U samoj formi bih želio samo izdvojiti jedna detalj :

```
<input type="hidden" name="idnovost" value="<?=$_GET["idnovost"]?>">
```

Za razliku od forme za unos novosti, ovog puta smo dodali novo polje za pohranu id novosti koju uređujemo. Za takve informacije koje se ključne za funkcioniranje skripte, ali ne želimo ostaviti mogućnost da ih korisnik izmjeni u procesu korištenja skripte obično koristimo hidden polja.

Nakon prikaza forme prestaje prvi korak naše skripte korisnik mora ispuniti i submitati formu da bi prešao na slijedeći korak u kojem se pohranjuju učinjene izmjene.

```
$sql="UPDATE novosti SET naslov='". $_POST["naslov"] ."', tekst='".$_POST["tekst"]
."' WHERE idnovost=". $_POST["idnovost"];

if (mysql_query($sql))

{

if (mysql_affected_rows() > 0 )

{

echo "Novost je uspješno uređena.";

} else {
```

```
echo "Novost nije izmjenjena.";

}

} else {

echo "Nastala je greška pri izmjeni novosti<br>" . mysql_error();

}
```

Prvi korak je dinamički popuniti SQL upit. Pošto je UPDATE upit već obrađen na njemu se nećemo zadržavati. Ostatak koda bi vam također već trebao biti poznat. Identičan je kao i kod izvršavanja DELETE upita. Nakon izvršavanja upita sa mysql_query() funckijom provjeravamo da li izmjenjen koji redak u tablici te na osnovi te informacije ispisujemo odgovarajuću poruku.

Ovdje bih želio istaknuti jednu zanimljivu stvar. U našem pregledniku otvorite uređivanje neke novosti te submitajte formu za uređivanje a da ne učinite nikakve izmjene na podacima novosti. Koja poruka vam se javila?

Trebala se javiti poruka „Novost nije izmjenjena“. O čemu se radi. Ukoliko izvršite UPDATE upit u kojem su novi podaci identični onima koji već postoje u tablici unutar tog retka (ili redaka) koji mijenjamo MySQL neće napraviti nikakve izmjene nad tim retkom (ili redcima) tako da mysql_affected_rows() vraća nulu kao rezultat.

Evo nas. Sa ovim primjerom smo završili sa osnovnim znanjem potrebnim za izradu web aplikacija pomoću MySql-a i PHP-a. Istina, znanje koje ste stekli nije dovoljno za neke kompleksnije aplikacije. No prošli smo kroz temelje izrade aplikacija ovakvog tipa. Ostalo nam je samo još bolje se upoznati sa SQL jezikom, posebno njegovim naprednijim opcijama i mogučnostima te vidjeti kako ubacivanje dodatnih, povezanih tablica utječe na izradu aplikacije. I time ćemo se baviti u ostatku vodiča. Ali, prije nego krenemo sa tim predložio bi da sami pokušate napraviti novi dio aplikacije iz gornjih primjera koji će služiti posjetiteljima sitea za čitanje novosti čiji smo administracijski dio upravo napravili.

PHP I FORME

Uvod u forme

Zbog prirode PHP-a kao serverskog skriptnog jezika za dinamičku manipulaciju web sadržaja koja u mnogim situacijama ovisi o korisničkom unosu ili korisničkim akcijama (klikanje na linkove i slične akcije), rad sa formama bi se dao svrstati u grupu bazičnih metoda prikupljanja informacija kroz web sučelje.

Forme se koriste u preko 90% svih manjih a posebno većih web aplikacija, neovisno u kojem je jeziku aplikacija napisana. Zbog ovoga je potrebno imati znanje samih HTML formi (tipove polja, koja su za koju namjenu najbolja, načine grupiranja elemenata …).

Uz poznavanje HTML dijela formi, potrebno je znati kako prihvatiti pomoću PHP-a vrijednosti koje su unesene u formu.

Ovaj članak se bavi tom problematikom od izrade jednostavnih formi i prihvaćanja informacija, do izrade kompliciranijih formi.
O formama općenito

Prije nego se bacimo na sočni dio izrade samih formi i prihvaćanje njihovih informacija malo ćemo se pozabaviti samim formama.

Počnimo sa vrlo jednostavnim pitanjem. Što je forma?

Sa tehničke strane gledano, forma je HTML element pomoću kojeg grupiramo više elemenata za unos podataka (tipa text box, select izbornik, opcije …). U te elemente korisnik može unijeti (ili izabrati) informacije koje se kasnije koriste za bilo koju svrhu (unos informacija u bazu podataka, slanje e-maila sa unesenim informacijama …).

Razlog grupiranja više elemenata za unos podataka u jednu formu je više – manje očit.

Uglavnom kada se vrši nekakav unos na web stranici, obično se radi o više povezanih informacija koje čine cjelinu (primjer : login forma sa usernameom i passwordom).

Pošto su elementi forme grupirani šalju se na obradu zajedno kada se forma 'submita'. (Ukoliko niste razumjeli posljednju rečenicu, nemojte se brinuti. Do kraja članka će vam sve biti jasno).

Sada kada znamo što je forma, idemo vidjeti kako staviti formu u HTML dokument.

Primjer 1 : Umetanje forme u HTML dokument

Otvorite svoj omiljeni HTML editor i upišite slijedeći kod. Dokument spremite pod imenom forma.htm i pogledajte ispis u svom omiljenom browseru.

```
<html>

<head>

<title>Primjer 1 : Umetanje forme u HTML dokument</title>

</head>

<body>

<h1>Primjer 1 : Umetanje forme </h1>

<hr>

<p>Ispunite formu : </p>

<form method="post" action="prihvat.php" name="forma1">

</form>

</body>

</html>
```

Primjer 1 : Objašnjenje

Kao što se vidi u ispisu, sama forma je nevidljiva. Nju možemo zamisliti kao nevidljivi 'kontejner' u kojeg se umeću novi elementi. Elemente ćemo dodavati malo kasnije. Prije toga bi želio prokomentirati par zanimljivih stvari u primjeru.

Prva stvar koja vam odmah upada u oči je ekstenzija našeg dokumenta forma.htm.

Pitate se, kako je moguće prihvatiti podatke iz forme ako se ne radi o PHP dokumentu. Razlog je dosta jednostavan. Forme se mogu nalaziti i u običnim html dokumentima. Bitno je samo da dokument koji prihvaća podatke forme bude PHP dokument, jer se u protivnom PHP kod za prihvat podataka forme neće moći obaviti.

Idemo sada pogledati sam HTML kod pomoću kojeg smo umetnuli formu u dokument.

```
<form method="post" action="prihvat.php" name="forma1">

</form>
```

Kao što vidite,. Radi se o najobičnijem HTML tagu koji ima svoj početak i kraj, te neke argumente. Argumenti su dosta bitni za cijelu priču, pošto određuju mnoge parametre o kojima ovisi tok događaja nakon submitanja (slanja) forme.

Prvi argument je method="post"

Ovo je vrlo bitan argument. On govori kojom metodom će se podaci iz forme proslijediti dokumentu (skripti) za obradu unesenih podataka. Metodama slanja podataka forme je posvećen cijeli slijedeći dio ovog članka, tako da ćemo za sada preći preko ovoga.

Drugi argument je action="prihvat.php"

Pomoću njega govorimo formi koji dokument se otvara kada se forma submita. Vrijednost argumenta može biti bilo kakav URL, uključujući i dokument na kojem se prikazuje forma sa jedne strane, te dokument na drugom siteu sa druge. O obradi forme u istom dokumentu na kojem se forma prikazuje će biti više govora kasnije u članku. Za sada je bitno znati da forma očekuje da se kod koji obrađuje njene informacije nalazi u dokumentu prihvat.php koji se nalazi u istom direktoriju kao i forma.htm

Treći navedeni argument je name="forma1"

On, očito, imenuje ovu formu. Ovaj argument nije bitan za prihvat i obradu podataka forme pomoću PHP-a. Važnost ovog argumenta se više veže uz izradu JavaScript klijentskih kodova, i služi za identificiranje ove forme. Pošto je tema ovog članka PHP i forme, nećemo ulaziti u detalje te problematike. Bitno je znati da se forma može imenovati, te da je u našoj situaciji ovaj argument moguće i izostaviti.

I to je to. Imamo formu bez elemenata za unos informacija koju ne možemo submitati pošto ne postoji submit gumb. Ali i to je nešto.

Metode slanja i prihvata podataka

U prošlom odsječku članka smo u primjeru rekli da je moguće odabrati metodu kojom će se podaci forme proslijediti dokumentu (skripti) za obradu njenih informacija. U ovom odsječku ćemo se detaljno pozabaviti ovom problematikom i napravit ćemo prvu cijelu formu, skupa sa prihvatom njenih podataka.

Ali idemo redom.

Dvije osnovne metode prosljeđivanja podatak forme nekom dokumentu (skripti) su POST i GET.

Odabir metode prosljeđivanja podataka forme se vrši navođenjem method argumenta <form> taga. Moguće vrijednosti method argumenta su 'post' i 'get'. Mislim da nije potrebno isticati koja vrijednost predstavlja koju metodu.

Odabirom metode utječemo na koji način će se sami podaci forme proslijediti stranici, i to je neovisno o jeziku u kojem će se ti podaci prihvatiti i obraditi. U PHP-u odabirom metode i odabiremo način na koji ćemo prihvatiti te podatke, no o tome malo kasnije.

Idemo vidjeti koje su točno te razlike u metodama pri samom prosljeđivanju informacija.

Prva razlika je ona najdrastičnija i odmah uočljiva.

Kada se odabere metoda GET podaci forme se šalju kroz komandnu liniju (query string, tj. iza znaka ? u adress baru browsera).

Odabirom metode POST podaci nisu vidljivi u komandnoj liniji već se šalju transparentno kroz header HTTP requesta. Detalje oko HTTP requesta i HTTP responsa neću pokriti u ovom tekstu pošto se radi o zasebnoj temi, ali ono što je bitno za znat je da se pri komunikaciji između klijenta i servera (browser – web server) šalju zahtjevi o odgovori koji se sastoje od više dijelova.

Kod requesta, ili zahtijeva za nekom stranicom od strane klijenta na nekom serveru se obično radi o čistom headeru zahtjeva koji u sebi sadrži adresu stranice koja se želi vidjeti, neke informacije o samom klijentu (tip browsera, OS …), te GET i POST podaci.

Kod responsa, ili uzvraćanja informacija servera klijentu se može raditi o više situacija. Ukoliko je nastala neka greška pri pribavljanju zatražene stranice onda se šalje samo response header sa kodom greške te nekim dodatnim informacijama (tip servera, protokol …). Ukoliko se requestom zatražila postojeća stranica onda se šalje response koji se sastoji od dva dijela. Od headera koji sadrži opet te generalne informacije skupa sa nekim dodacima, poput COOKIESA, te od body-a koji u sebi obično sadrži HTML kod stranice koju smo zatražili.

Dobro, kakve sve ovo ima veze sa POST-om i GET-om. Kao što sam rekao, POST-om se podaci forme šalju kroz request header i time se na njih ne može utjecati izmjenom linka u adress baru browsera i shvaćanje procesa komunikacije između servera i klijenta nam može kasnije dobro doći.

Također, postoji i ograničenje na količinu podataka koji se mogu poslati putem GET metode. Ograničenje ovisi o postavkama samog servera na kojem se nalazi dokument na kojem se procesira forma i varira od 256 byte-ova do čak 32 Kb. Zato, ukoliko se radi o formama koje u sebi sadržavaju polja za unos velikih tekstova ili sličnih podataka GET i nije najsretnija metoda za prosljeđivanje informacija.

Kako se odlučiti koju metodu koristiti? Metoda koja se koristi za prosljeđivanje informacija ovisi o situaciji u kojoj se nalazite. Svaka metoda ima svoje vrline i mane.

Rekao sam da se u GET metodi informacije lijepe na sam URL, što ovu metodu čini idealnom u slučaju kada želite omogućiti posjetiteljima sitea da spreme stranicu koju gledaju u svoje favoritese, jer će spremiti URL zajedno sa zalijepljenim query stringom. Primjer ovoga bi bio Google i njihova tražilica koja sve podatke iz formi lijepi u query string baš iz tog razloga.

Sa druge strane GET je vrlo nesigurna metoda jer ju posjetitelj vrlo lako može izmijeniti jednostavnom izmjenom URL-a u adress baru svog browsera, tako da nije preporučljivo koristiti ovu metodu za prosljeđivanje recimo usernamea i passworda u login formama i sličnih osjetljivih informacija.

Ali neka pravila u odabiru metode ne postoje i kao programeri aplikacija odlučujemo koju ćemo metodu koristiti, ovisno o situaciji u kojoj se nalazimo, i ovo je izvan aspekta ovog članka. Jedino pravilo koje se mora poštovati je da se obavezno mora koristiti POST metoda kada se izrađuju upload forme, ali o njima nećemo sada govoriti.

Idemo sada vidjeti sve gore spomenute implikacije o metodama na praktičnom primjeru.

Primjer 2 se sastoji od dva dijela. Radi se o istoj formi, samo što se u slučaju a) koristi GET metoda, a u slučaju b) POST.

Primjer se sastoji od dva dokumenta za oba slučaja. Prvi ćemo nazvati forma.php, a drugi prihvat.php. Iz imena zaključujete da će se u forma.php nalaziti sama forma, a u prihvat.php će se nalaziti PHP skripta koja će prihvatiti te informacije i ispisati ih na ekran. Ovi primjeri bi trebali ilustrirati osnovnu metodu za prihvat podataka iz forme.

Primjer 2 a) : Jednostavna forma pomoću GET metode

Dokument : forma.php

```
<html>

<head>

<title>Primjer 2 a) : Prosljeđivanje informacije GET metodom</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

```
</head>

<body>

<h2>Primjer 2 a) : Prosljeđivanje informacije GET metodom </h2>

<hr>

<form name="primjer2-a" method="get" action="prihvat.php">

Unesi svoje ime :

<input name="ime" type="text" >

<br>

<input name="submit" type="submit" value="Pošalji">

</form>

</body>

</html>
```

Dokument : prihvat.php

```
<html>

<head>

<title>Primjer 2 a) : Prihvat podataka GET metode</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<?

echo "Pozdrav " . $_GET["ime"];

?>

</body>

</html>
```

Pokrenite forma.php i ispunite polje sa svojim imenom te kliknite na gumb na kojem piše Pošalji. To bi trebali izgledati otprilike ovako :

Nakon pritiska gumba, tj. submitanja forme se u prozoru otvara dokument prikaz.php koji ispisuje uneseno ime nazad na ekran. To izgleda otprilike ovako :

Primjer 2 a) objašnjenje :

Idemo prvo pogledati bitne dijelove forma.php dokumenta.

U prvom primjeru smo prošli kroz dijelove <form> taga tako da ću ih sada preskočiti. Potrebno je uočiti samo da je u method argumentu namještena vrijednost get.

Za razliku od prvog primjera ovog puta smo u formu ubacili dva elementa. Radi se o <input> tagu koji ima vise različitih tipova, sa kojima ćemo se upoznati malo kasnije. Prvi put je korišten u svom text tipu :

```
<input name="ime" type="text" >
```

a drugi put se radi o gumbu :

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

U oba elementa je potrebno primijeniti korištenje name argumenta. Pomoću njega dajemo polju ime i to ime koristimo u PHP-u za prihvat podataka koji su uneseni u to polje.

Ukoliko ne imenujete polje, podaci koji se unesu u njega nisu vidljivi PHP skripti koja je namijenjena za prihvat podataka iz te forme.

Oko detalja svih različitih elemenata forme, te načinima rada sa njima će biti više govora kasnije u članku. Ovaj primjer samo ilustrira kako prihvatiti podatke iz forme.

Prihvat podataka je sam po sebi vrlo jednostavan. Unutar PHP-u postoje superglobalne varijable. To su varijable (nizovnog tipa) koje definira sam PHP pri pokretanju svake skripte. Superglobalne varijable nisu tema ovog članka, tako da ćemo se mi samo pozabaviti sa dvije od njih.

Prva je $_GET i ona je korištena u prihvat.php dokumentu našeg primjera.

```
echo "Pozdrav " . $_GET["ime"];
```

Gornja linija ispisuje poruku koja se sastoji od konstantnog dijela Pozdrav i dijela u kojem se ispisuje ono što je uneseno u text box polje u forma.php dokumentu koje smo nazvali ime.

Iz ovog jednostavnog primjera je vidljivo da se podacima iz forme pristupa pomoću odgovarajuće superglobalne varijable. Vrijednost nekog elementa forme se nalazi unutar te superglobalne varijable na indeksu koji odgovara imenu tog elementa forme. U kojoj se superglobalnoj varijabli nalaze podaci ovisi o izabranoj metodi prosljeđivanja podataka. U ovom primjeru je metoda bila GET, tako da je korištena superglobalna varijabla $_GET.

U b dijelu ovog primjera ćemo koristiti POST metodu, tako da će korištena superglobalna varijabla biti $_POST.

Prije nego što krenemo na drugi dio primjera, primijetite da su se podaci koji smo ispunili u forma.php dokumentu zalijepili na link u prihvat.php dokumentu koji ispisuje te podatke.

```
http://localhost/phpbook/forme/primjer2-a /prihvat.php?ime=Marko&submit=Po%9Aalji
```

Sve iza znaka ? u gornjem linku se naziva QUERY STRING.

Radi se o seriji parova imena i vrijednosti međusobno odvojenim znakom &.

Ako si sada pomišljate da bi bilo moguće ovim načinom prenijeti podatke iz jedne stranice na drugu čisto putem linkova, onda dobro mislite. Naime, PHP-u je svejedno da li se podaci u query stringu

popunjavaju putem neke forme, ili se radilo o <a> tagu u kojem je u href argumentu napisan url skupa sa query stringom.

Znači, ovo bi bio ekvivalent našoj formi iz primjera, kada se u text box unese Marko

```
<a href='prihvat.php?ime=Marko'>Moje ime je Marko</a>
```

Čak smo i izostavili drugu varijablu u query stringu pošto nam ona za sada nije potrebna u prihvat.php dokumentu.

Idemo sada ovom primjeru promijeniti metodu u POST i vidjeti kako to mijenja način prihvata podataka.

Primjer 2 a) : Jednostavna forma pomoću POST metode

Dokument : forma.php

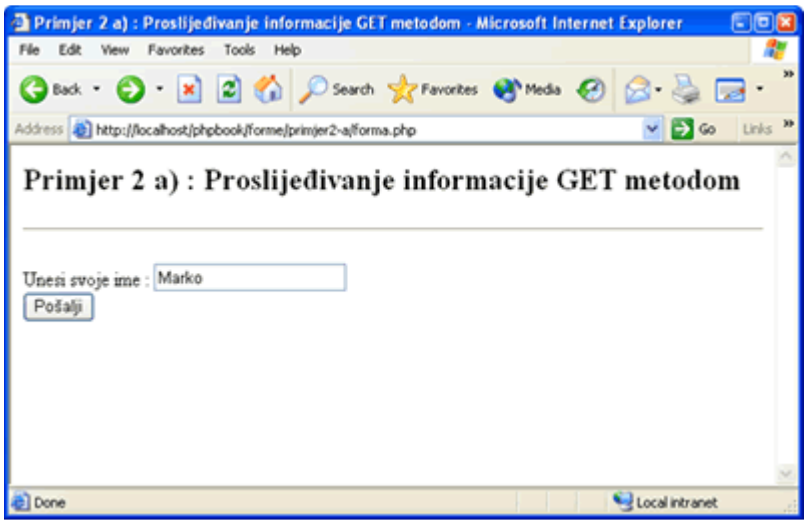
```
<html>

<head>

<title>Primjer 2 b) : Prosljeđivanje informacije POST metodom</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h2>Primjer 2 b) : Prosljeđivanje informacije POST metodom </h2>

<hr>

<form name="primjer2-b" method="post" action="prihvat.php">

Unesi svoje ime :

<input name="ime" type="text" >

<br>

<input name="submit" type="submit" value="Pošalji">

</form>

</body>

</html>
```

Dokument : prihvat.php

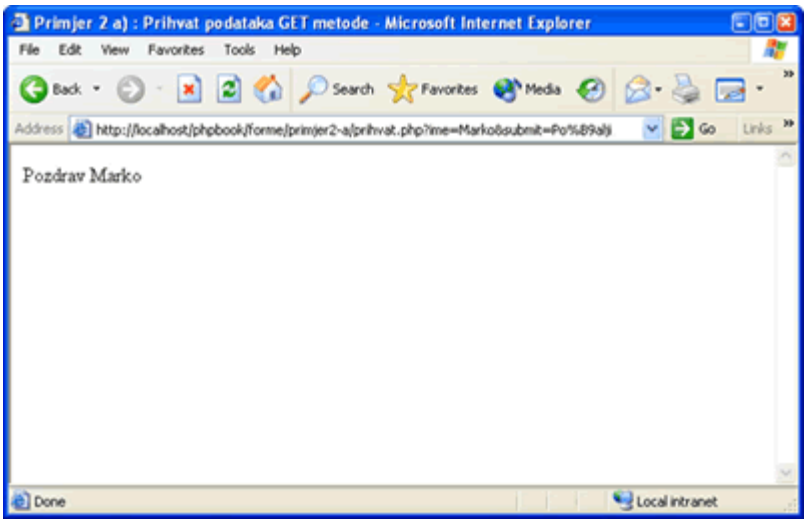
```
<html>
```

```
<head>

<title>Primjer 2 b) : Prihvat podataka POST metode</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<?

echo "Pozdrav " . $_POST["ime"];

?>

</body>

</html>
```

Pokrenite forma.php i ispunite polje sa svojim imenom te kliknite na gumb na kojem piše Pošalji. To bi trebali izgledati otprilike ovako :



Nakon pritiska gumba, tj. submitanja forme se u prozoru otvara dokument prikaz.php koji ispisuje uneseno ime nazad na ekran. To izgleda otprilike ovako :



Primjer 2 b) objašnjenje :

Princip prihvata podataka polsanih POST metodom je isti kao i kod prihvata podataka GET metode, sa manjim razlikama.

Prva razlika je u forma.php dokumentu, gdje je promijenjena metoda u POST.

<form name="primjer2-b" method="post" action="prihvat.php">

Druga razlika je u već spomenutoj superglobalnoj varijabli koja je ovog puta $_POST.

```
echo "Pozdrav " . $_POST["ime"];
```

Dodatna i očita razlika je da sada više nema query stringa i nije moguće direktno utjecati na podatke nakon što je forma ispunjena i submitana.

Prije nego se dalje nastavimo upoznavati sa elementima formi želio bih istaknuti par vrlo važnih stvari.
Register globals

U oba dijela primjera 2 smo za prihvat podataka koristili superglobalne varijable. No to nije uvijek obavezno. Naime, kada je u konfiguraciji PHP-a namještena direktiva register_globals na On onda je podacima iz forme ili query stringa moguće pristupiti i izostavljanjem imena superglobalnih varijabli, te podacima iz elemnta forme pristupamo pomoću varijable ekvivalentnog imena elementa forme čijoj vrijednosti želimo pristupiti.

Da malo ovo razjasnim na primjeru.

U oba slučaja iz primjera 2 smo mogli koristiti prihvat.php dokument koji bi izgledao ovako :

```
<html>

<head>

<title>Primjer 2 c) : Prihvat podataka</title>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<?

echo "Pozdrav " . $ime;

?>

</body>

</html>
```

Znači, umjesto $_POST['ime'] ili $_GET['ime'] smo koristili samo $ime.

Iako ovo na prvi pogled izgleda kao bolji način za prihvat podataka ipak će vam mnogi iskusni PHP programeri reći da to nije istina.

Naime, ovom metodom ste si napravili pravu malu sigurnosnu crnu rupu. Radi se o tome da ukoliko je PHP konfiguriran sa register_globals=On svatko može vašoj skripti promijeniti vrijednost neke varijable čistim navođenjem te varijable u query string.

Također, ponekada je poželjno znati da li neka varijabla koju u skripti koristimo spada u session varijable, cookije ili je došla iz forme ili query stringa.

Iz tog razloga je poželjno držati register_globals na Off te koristiti superglobalne varijable za prihvat željenih podataka.
Provjera metode pristupa dokumentu / skripti

U nekim situacijama je potrebno znati kojom metodom je neki dokument / skripta otvorena, te na osnovi toga obavljamo potrebne operacije.

Ovo je moguće provjerom vrijednosti $_SERVER["REQUEST_METHOD"] ili skraćeno $REQUEST_METHOD varijable.

Njene moguće vrijednosti su POST i GET, ovisno o metodi otvaranja stranice. Njena vrijednost će uvijek biti GET osim u slučaju da se dokumentu pristupilo nakon submitanja forme sa POST metodom.

Iz tog razloga ovo nije najbolja metoda za provjeru da li je netko stvarno ispunio formu i pristupio dokumentu za prihvat i obradu podataka. O toj provjeri će biti govora malo kasnije kada ćemo napraviti obradu forme sa više submit gumba.

No prije toga, da vam demonstriram jednu zanimljivu činjenicu. Otvorite formu slučaja b Primjera 2 te ju ispunite i submitajte. Kada se otvori prihvat.php refreshajte stranicu. Sada bi vas browser trebao pitati da li želite opet poslati podatke forme stranici te morate odabrati potvrdnu akciju ukoliko želite opet vidjeti prihvat.php. U slučaju da odaberete da ne želite opet poslati podatke dobijete error page (iako ovo ovisi o konfiguraciji samog browsera).

Sada pokušajte ovo isto učiniti za a slučaj Primjera 2. Kod njega vas browser ništa ne pita već odmah prikazuje stranicu.

Zbog ovoga je GET metoda puno praktičnija kada se rade forme za pretraživanje, jer svako dodatno klikanje pri surfanju kroz site samo može iznervirati posjetitelja, što nam nije cilj, zar ne?

Također, kod search formi je dodatna, već spomenuta, prednost ta što se query string sprema zajedno sa URL-om u favoritese surfera, tako da se moguće opet vratiti na isto pretraživanje nakon više dana jednostavnim odabirom bookmarka.

Ova činjenica također dobro dođe kod debugiranja skripti za prihvat podataka forme. Ukoliko postoji greška u prihvat.php dokumentu možete ju ispraviti u editoru u kojem pišete skriptu, te nakon što sačuvate izmjene kliknete refresh u browseru i dokument će se prikazati ukoliko ste ispravili sve greške. Radi toga se nije potrebno vratiti na formu te ju ponovno ispuniti da bi testirali da li su sve greške ispravljene.
Forme i hrvatski znakovi

Hrvatski znakovi su sudeći po mnogim forumima i raspravama na usenetu dosta velik problem, posebno početnicima. Iz tog razloga bih želio malo rasvijetliti ovu problematiku.

Problem neprikazivanja, tj. lošeg prikazivanja hrvatskih znakova leži u nepravilnom i nedosljednom korištenju charseta unutar svih dokumenata skripte.

Naime, prije izrade skripti dobra praksa je odlučiti se za jedan charset te ga dosljedno koristiti na svim dokumentima unutar sitea i time ćete izbjeći sve probleme sa njima. Ovo pravilo vrijedi i za rad sa bazom podataka.

Da se uvjerite na primjeru.

Otvorite prihvat.php iz jednog od slučaja Primjera 2, te mu promijenite

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

u

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
```

Zatim otvorite forma.php iz istog slučaja, ispunite formu sa nekim hrvatskim znakovima te ju submitajte.

Kao što vidite, neki od znakova se nisu prikazali pravilno. Ako uskladite charsete hrvatski znakovi će se prikazivati bez problema.

Ovo su bile osnove slanja i prihvata podataka iz forme. U slijedećem dijelu članka će biti govora o pojedinim elementima HTML formi, te o načinima rada sa njima. No, prije toga ćemo malo zakoračiti u područje logike prikaza formi radi pojednostavljena primjera koji dolaze u sljedećem dijelu.

Prikaz i obrada unutar jednog dokumenta

Ovo je dio teme logike prikaza i obrade formi, koja će biti detaljno pokrivena pri kraju ovog članka. Iz tog razloga ću zaobići veći dio objašnjenja prednosti i mana ove metode prikaza i obrade formi. Za sada je bitno samo da shvatite princip po kojem će se forma prikazivati i obrađivati, jer ćemo kasnije u članku malo razviti ovaj princip i učiniti ga puno 'pametnijim'.

Idemo vidjeti o čemu se radi.

U srednjim i većim aplikacijama je vrlo nepraktično imati odvojene dokumente za prikaz i obradu forme iz očitog razloga što bi se broj dokumenata nekog sitea popeo do tako velikog broja da bi postao problem praćenja što se gdje odvija unutar strukture dokumenata sitea, te si takvom praksom samo bespotrebno kompliciramo izradu sitea tj. aplikacije.

Iz tog razloga je poželjno stvari koje su usko povezane, poput prikaza i obrada forme držati unutar istog dokumenta u najmanju ruku. O ponešto naprednijim metodama logike prikaza i obrade forme će biti govora pri kraju ovog članka.

Idemo pogledati konkretan primjer pa ćemo ga prokomentirati.

U primjeru ćemo koristiti istu formu i obradu iz b slučaja Primjera 2.

Primjer 3 : Prikaz i obrada forme unutar jednog dokumenta

```
<html>

<head>

<title>Primjer 3 :Prikaz i obrada forme unutar jednog dokumenta</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h2>Primjer 3 :Prikaz i obrada forme unutar jednog dokumenta</h2>

<hr>

<?

if ( ! $_POST["submitaj"] ){

?>

<form name="primjer3" method="post" action="<?=$PHP_SELF?>">

Unesi svoje ime :

<input name="ime" type="text" >

<br>

<input name="submitaj" type="submit" value="Pošalji">

</form>

<?

} else {

echo "Pozdrav " . $_POST["ime"];
```

```
}

?>

</body>

</html>
```

Objašnjenje :

Ovaj primjer je ništa više nego malo presložen Primjer 2, i to njegov b slučaj.

Zapravo smo eliminirali potrebu za dva odvojena dokumenta i sve se obavlja unutar istog dokumenta. Ovo bi se usudio nazvati prvim stupnjem strukturiranja koda formi. Prvi zato što je ovo zapravo tek prvi od više koraka u izradi naprednih i pametnih form sistema, o kojima će biti više govora kasnije u članku.

Idemo sada proći kroz novine u ovom primjeru.

Prva stvar koju odmah uočavate je

```
if ( ! $_POST["submitaj"] ){
```

Ova linija provjerava da li postoji neka vrijednost u varijabli $_POST["submitaj"]. Ako se sjećate, već smo radili ponešto sličnu provjeru. Mislim na provjeru kojom metodom je prestupljena stranica pomoću provjere $REQUEST_METHOD varijable.

Ta provjera i nije najbolja za našu situaciju. Razlog tome je štou toj varijabli uvijek postoji neka vrijednost, po defaultu GET, iako ne postoji query string , tako da njena vrijednost ne garantira da je netko stvarno ispunio formu i došao na stranicu.

U našem primjeru provjeravamo postojanje vrijednosti unutar superglobalne $_POST varijable, a ukoliko se u njoj nalazi neka vrijednost podrazumijeva i da je $REQUEST_METHOD == POST.

Ono što treba primijetiti u primjeru je da se provjerava postojanje vrijednosti za submit gumb unutar forme. To radimo zato što je ta vrijednost obavezno prisutna ukoliko se forma ispunila i submitala, zato što smo joj u samom HTML-u dali vrijednost pomoću value argumenta u kodu submit gumba.

Želio bih napomenuti da u ovoj logici ima smisla provjeravati samo vrijednosti za submit gumbe jer je jedino njima garantirano postojanje vrijednosti. Recimo, u primjeru nema smisla provjeravati postojanje vrijednosti u $_POST["ime"] kao uvijet prikaza ili obrade forme, jer surfer možda ne ispuni to polje, te bi mu se nakon submitanja forme nanovo prikazala forma ukoliko bi ostavio polje ime prazno.

Postoji još jedna vrlo zanimljiva i korisna stvar oko submit gumba, no na nju ću se vratiti malo kasnije, u Primjeru 4.

Vratimo se mi na naš primjer.

Logika primjera je poprilično jednostavna. Ukoliko ne postoji vrijednost u $_POST["submitaj"] znači da forma nije još ispunjena i trebaju prikazati, a ukoliko ta vrijednost postoji znači da je forma ispunjena i submitana, te želimo obraditi te podatke.

U samoj formi je dodana još jedna novina(boldano u liniji ispod)

```
<form name="primjer3" method="post" action="<?=$PHP_SELF?>">
```

$PHP_SELF je varijabla koju definira sam PHP i u njoj se nalazi ime trenutno aktivnog dokumenta. Pošto mi želimo da se forma prikaže i obradi unutar istog dokumenta, to je točno ono što nama treba. Iako smo tu mogli ručno upisati ime našeg dokumenta, korištenje $PHP_SELF je preporučljivo, zato što ovako možemo promijeniti ime našeg dokumenta, a u njemu će sve i dalje raditi kako je zamišljeno, zato što PHP sam namješta njenu vrijednost.

Kao što sam rekao, ovo je vrlo jednostavan model prikaza i obrade formi, i on je idealan za login forme, forme za unos i izmjenu podataka. Postoje neki problemi za koje ovo nije idelano rješenje, posebno u administracijskim alatima, no o tim problemima ću više govoriti kada bude bilo govora o logikama prikaza formi.

Sada ću se vratiti na onu zanimljivu stvar oko submit gubma. Radi se o slučaju kada želimo imati u jednoj formi više submit gumba, te ovisno o tome koji je kliknuti želimo u obradi forme obaviti različite operacije.

Razlozi za ovako nečim su vrlo česti, pogotovo kada se radi neki administracijski alatima, gdje je potrebno moći napraviti više različitih operacija nad istim podacima (recimo želimo omogućiti brisanje većeg broja novosti u našoj news skripti, ali želimo imati i gumb koji bi obrisao sve poruke odjednom).

Idemo ovo promotriti na konkretnom primjeru. Primjer se sastoji od forme sa dva submit gumba, te od obrade te forme unutar istog dokumenta. U obradi forme ćemo ispisati poruku koji je gumb stisnut.

Primjer 4: Forme sa više gumba

```
<html>

<head>

<title>Primjer 4 : Forma sa više submit gumba</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h2>Primjer 4 :Forma sa više submit gumba</h2>

<hr>

<?

if ( $REQUEST_METHOD != "POST"){

?>

<form name="primjer4" method="post" action="<?=$PHP_SELF?>">

<input type="submit" name="gumb1" value="Gumb 1">

<br>
```

```
<input type="submit" name="gumb2" value="Gumb 2">

</form>

<?

} else {

// obrada forme

if ($_POST["gumb1"])

echo "Pritisnuli ste gumb 1";

if ($_POST["gumb2"])

echo "Pritisnuli ste gumb 2";

}

?>

</body>

</html>
```

Ovog puta ćemo preskočiti slike već idemo odmah na objašnjenje.

Objašnjenje :

Ovog puta smo kao uvjet prikaza ili obrade forme koristili $REQUEST_METHOD. On je korišten zato što ovog puta postoje dva submit gumba, i stiskanjem bilo kojeg od njih želimo umjesto prikaza forme pokrenuti obradu iste.

Kvaka leži u tome (a to je i cijeli point ovog primjera) da će se nakon sumbitanja forme pomoću jednog od tih gumba popuniti vrijednost unutar superglobalne $_POST varijable samo onog gumba koji je pritisnut, što dokazuje dio koda koji obrađuje formu.

```
if ($_POST["gumb1"])

echo "Pritisnuli ste gumb 1";

if ($_POST["gumb2"])

echo "Pritisnuli ste gumb 2";
```

Zato što mi u trenutku u kojem odlučujemo da li se treba prikazati forma ili se treba obraditi forma ne znamo koji je gumb pritisnut moramo koristiti $REQUEST_METHOD varijablu kao provjeru da li je forma već ispunjena. Napominjem, ukoliko je metoda forme GET primjer će sa kodom iz primjera (uz učinjene preinake iz POST u GET na svim mjestima) uvijek obavljati obradu forme pošto se u $REQUEST_METHOD uvijek nalazi GET, iako ne postoje podaci u query stringu.

Normalno, drugo rješenje problema je da se u if-u koji odlučuje da li se prikazuje ili obraduje forma ispitamo stanje svih gumba unutar forme. Nešto poput ovoga :

```
if ( !( $_POST["gumb1"] or $_POST["gumb2"] ) ) {
```

Normalno, uočavate da bi sa ovom shemom morali navesti sve postojeće gumbe da osiguramo pravilno ponašanje našeg malenog sistema prikaza i obrade forme.

U formama sa više gumba se obično koristi malo drugačiji model prikaza i obrade forme, te ću se njemu posvetiti malo kasnije u ovom članku. Za sada je bitno da znate da se PHP-u prosljeđuju vrijednosti samo onih submit gumba koji su pritisnuti pri submitanju forme.

Idemo se sada na trenutak vratiti na metode slanja formi, te ćemo se nakon toga baciti na pojedine elemente forme te na rad sa njima.

Kombiniranje POST i GET metode prosljeđivanja podataka kroz jednu formu

Ponekada je potrebno kombinirati obje metode prosljeđivanja podataka.

Recimo, u slučaju kada se unutar jednog PHP dokumenta nalazi više 'dijelova' sitea

Tu govorim o tzv. Višenamjenskim stranicama. Ukratko, koncept takvih dokumenata je da se na osnovi jedne (ili više) varijable iz query stringa surferu prikazuju potpuno različiti dijelovi našeg sitea.

Recimo, ako surfer otvori link tipa http://www.moj-site.com/indeks.php?nav=novosti

Prikazuje mu se lista svih novosti, a ako otvori link

http://www.moj-site.com/indeks.php?nav=kontakt otvara mu se kontakt forma. Očito je da u ovom slučaju moramo nekako proslijediti $_GET['nav'] varijablu kroz query string i kada se popuni kontakt forma da bi sve radilo kako spada.

Ovo je moguće ukoliko se radi o formi sa POST metodom slanja podataka.

Idemo pogledati primjer pa će stvari biti jasnije. Sam primjer je malo kompleksniji radi ilustracije konkretnog problema. Radi se o višenamjenskoj stranici čiji jedan dio sadrži formu koju ćemo prikazivati i obrađivati.

Primjer 5: Kombiniranje POST i GET metode u istoj formi

```
<html>

<head>

<title>Primjer 5: Kombiniranje POST i GET metode prosljeđivanja podataka</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h2>Primjer 5 : Kombiniranje POST i GET metode prosljeđivanja podataka</h2>
```

```
<hr>
<p>Dobrodošli na višenamjensku stranicu:<br>
<br>
Dijelovi stranice :</p>
<ul>
<li> <a href="<?=$PHP_SELF?>">Početna stranica</a></li>
<li><a href="<?=$PHP_SELF?>?nav=tekst">Stranica sa tesktom</a></li>
<li><a href="<?=$PHP_SELF?>?nav=forma">Forma</a></li>
</ul>
<hr>
<?
// realizacija višenamjenske stranice
switch (@ $_GET["nav"]){
case "forma":
// dio dokumenta sa prikazom i obradom forme
if (! $_POST["SBgumb"]){
?>
<form name="primjer3" method="post" action="<?=$PHP_SELF?>?nav=forma">
Unesi svoje ime :
<input name="ime" type="text" >
<br>
<input name="SBgumb" type="submit" value="Pošalji">
</form>
<?
} else {
// obrada forme
echo "Polje ime sadrži vrijednost : " . $_POST["ime"];
}
```

```
break;

case "tekst":

?>

<p>Ovo je dio stranice sa

nekim većim tekstom koji služi samo za ilustraciju </p>

<?

break;

default :

// obavlja se ako ne postoji $_GET["nav"] varijabla ili ako ima nenavedenu
vrijednost

// prikazuje se pri dolasku na dokument

echo "Nalazite se na početnoj stranici";

}

?>

</body>

</html>
```

Objašnjenje :

Već sam spomenuo da se ovdje radi o višenamjenskoj stranici koja je realizirana pomoću kontrolne varijable $_GET["nav"]. Ovisno u vrijednosti u njoj naš dokument surferu vraća drugačiji ispis. Pošto tema ovog članka nisu višenamjenske stranice dio sa kojim ćemo se detaljnije pozabaviti je slučaj kada se u $_GET["nav"] nalazi vrijednost forma kada se prikazuje naša forma.

```
if (! $_POST["SBgumb"]){

?>

<form name="primjer3" method="post" action="<?=$PHP_SELF?>?nav=forma">

Unesi svoje ime :

<input name="ime" type="text" >

<br>

<input name="SBgumb" type="submit" value="Pošalji">

</form>

<?
```

```
} else {

// obrada forme

echo "Polje ime sadrži vrijednost : " . $_POST["ime"];

}
```

U ovom isječku koda koji prikazuje i obrađuje formu je boldan dio na kojem ćemo se malo zadržati.

Koncept je poprilično jednostavan i razumljiv, no postoje neke stvari na koje je potrebno paziti kod ovog načina prosljeđivanja podataka. No idemo redom.

U prvom primjeru ovog članka sam rekao da action argument <form> taga služi za određivanje URL-a do dokumenta koji očekuje podatke iz te forme te ih obrađuje i radi sa njima ono što je potrebno. Pošto je u ovom slučaju dokument koji prihvaća podatke isti kao i onaj koji prikazuje formu kao prvi dio linka koristimo $PHP_SELF varijablu radi osiguravanja da će se opet otvoriti isti dokument. No, u ovom slučaju je problematika dodatno zakomplicirana time što se forma nalazi unutar višenamjenske stranice i ukoliko nekako ne proslijedimo vrijednost i za $_GET["nav"] varijablu nakon submitanja forme će se umjesto koda za obradu forme pokrenuti default grana switch strukture.

Iz tog razloga smo ručno 'zalijepili' na kraj action argumenta i query string sa vrijednosti za $_GET["nav"] varijablu, tako da osiguramo da će se nakon submitanja forme opet pokrenuti pravilna grana switch-a (ona u kojoj se nalazi kod za prikaz i obradu forme).

Pošto smo proslijedili tu vrijednost naša skripta će nakon submitanja obaviti obradu forme. Princip prikaza i obrade forme je isti onome iz primjera 3 tako da se ovog puta neću zadržavati na njegovom objašnjenju.

Kao što sam rekao, u ovom konceptu je potrebno paziti na neke stvari da bi sve radilo. Prva i očita stvar je da moramo proslijediti točnu vrijednost za $_GET["nav"]. No druga, ne toliko očita stvar je da ovaj koncept ne radi ukoliko se radi o formi sa GET metodom.

Problem leži u samom HTML-u, tj HTTP protokolu. Naime, kada se formi da GET metoda i u action argument te forme ručno upišemo link zajedno sa query stringom, taj query string će biti automatski zamijenjen sa imenima i vrijednostima elemenata iz forme. Ilustracije radi, pokušajte pokrenuti gornji primjer, samo promijenite metodu slanja forme iz POST u GET.

No, stvari ipak nisu toliko crne. Postoji način kako 'natjerati' formu da ipak proslijedi i $_GET["nav"] varijabli nakon submitanja forme ukoliko se forma šalje GET metodom. Jedan od načina je da u formu dodamo dodatno, nevidljivo, polje imena nav te u njega spremimo pravilnu vrijednost.

Znači, da bi stvari radile sa GET metodom, u gornjem primjeru treba samo izmijeniti HTML forme sa slijedećim kodom :

```
if (! $_GET["SBgumb"]){

?>

<form name="primjer3" method="post" action="<?=$PHP_SELF?>">

<input name="nav" type="hidden" value="forma">

Unesi svoje ime :

<input name="ime" type="text" >
```

```
<br>

<input name="SBgumb" type="submit" value="Pošalji">

</form>

<?

} else {

// obrada forme

echo "Polje ime sadrži vrijednost : " . $_GET["ime"];

}
```

Ovim primjerom ćemo završiti prvi dio ovog članka u kojem se govorilo o nekim osnovnim konceptima oko rada sa formama. Vidjeli smo na koje sve načine se mogu podaci proslijediti iz forme nekoj PHP skripti, te kako u toj skripti prihvatiti te podatke i izvršiti neke operacije nad njima. Za sada smo samo ispisivali podatke nazad 'na ekran', ali u zaključnom dijelu članka ćemo malo proširiti i taj dio.

No, prije toga ćemo proći kroz sve elemente koji se mogu naći u formi, te ću ilustrirati neke metode rada sa tim različitim vrstama elemenata.

Elementi za unos podataka

Kao što smo do sada u primjerima vidjeli, sve forme se sastoje od dva osnovna dijela. Prvi dio je sam <form> tag pomoću kojeg logički povezujemo više elemenata za unos / izbor podataka. Normalno, ti elementi bi onda bili drugi dio svake forme, i jedno bez drugoga ne ide.

Pošto postoji vrlo očita potreba za više različitih načina prikupljanja podataka od korisnika postoje različiti elementi od kojih svaki ima neke svoje osobine. Neki elementi imaju mogućnost direktnog unosa podataka od strane surfera, dok mu drugi omogućuju da izabere jednu od ponuđenih vrijednosti, neke vidi u formi a neke ne. Sada ćemo se upoznati sa svim tim različitim elementima i njihovim karakteristikama, te ćemo vidjeti neke načine za olakšavanje posla nama kao programerima tih formi.

Do sada smo u primjerima koristili samo text box element, radi jednostavnosti primjera. Idemo prvo vidjeti koji sve elementi postoje pa ćemo se malo zadržati na svakom od njih.

Elementi HTML formi :

- Text box
- Text area
- Sakriveno polje / hidden field
- Checkbox
- Radio gumb
- Lista / meni – select izbornik
- Polje za upload datoteka
- Gumbi

U slijedećim opisima ću zanemariti mnoge dijelove samog HTML-a poput davanja širine i visine polja i sličnih stvari, pošto te stvari nisu bitne za svladavanje rada sa formama u PHP-u, već ću se

koncentrirati samo na ključne dijelove. Ukoliko želite saznati više o samom HTML dijelu ove priče preporučam da posjetite www.w3schools.com ili neki slični site, te tamo proučite detalje HTML-a formi.

Text box polje

```
<input type="text" name="tekst">
```

Ovo je element sa kojim smo se već upoznali, i mislim da ga nije potrebno previše komentirati. Radi se o polju za unos jedne linije teksta. Ime pomoću kojeg mu pristupamo u PHP-u mu navodimo u name atributu.

Ukoliko želimo da se pri učitavanju stranice u ovom elementu nalazi neka vrijednost možemo ju navesti pomoću value atributa.

```
<input type="text" name="tekst" value="Ovo je defaultna vrijednost polja i može se promijeniti">
```

Sa value atributom ćemo se detaljnije baviti u kasnijim dijelovima članka, kada ću pričati o strukturiranju koda za prikaz / obradu formi. Za sada je bitnbo samo da znate da je moguće HTML-om popuniti element forme.

Pošto sam u dosadašnjim primjerima uvijek koristio ovo polje, ovdje ću preskočiti primjer sa prihvatom podataka iz tog elementa te ćemo krenuti dalje.
Text area

```
<textarea name="velikiTekst"></textarea>
```

Ovo je polje za unos texta od više redova. Znači, korisnik može prebaciti unos teksta u novi red te ga kao takvog poslati na obradu.

Pravila oko imenovanja su mu ista kao i za sve ostale elemente.

Za razliku od text boxa, text area nema value atribut, već mu se defaultna vrijednost zadaje na slijedeći način.

```
<textarea name="velikiTekst">
Ovo je defaulta vrijednost elementa i može imati više redova</textarea>
```

Kod davanja te defaultne vrijednosti je bitno paziti da će se 2 space znaka u samom elementu prikazati kao dva prazna prostora, te da prelazak u novi red u HTML-u uzrokuje prebacivanje unosa podataka u elementu u novi red.

Idemo pogledati na primjeru kako koristiti text area element.

Primjer 6 : Korištenje text area elementa

```
<html>

<head>

<title>Primjer 6 : Korištenje text area elementa</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
```

```
</head>

<body>

<h2>Primjer 6 : Korištenje text area elementa</h2>

<hr>

<?

if ( ! $_POST["submitaj"] ){

?>

<form name="primjer3" method="post" action="<?=$PHP_SELF?>">

Unesi tekst od više redova :<br>

<textarea name="tekst"></textarea>

<br>

<input name="submitaj" type="submit" value="Pošalji">

</form>

<?

} else {

echo "<b>Unijeli ste : </b><br>" . $_POST["tekst"];

echo "<hr>";

// ispis sa paženjem na prelazak u nov red.

echo "<b>Tekst u izvornom obliku : </b><br>";

echo nl2br($_POST["tekst"]);

}

?>

</body>

</html>
```

Objašnjenje :

Kao što primjećujete, logika prikaza i obade forme je ona iz primjera 3 i ona će biti korištena u većini primjera u ovom dijelu članka.

Još jedna stvar koju možete odmah primijetiti je i način na koji smo pristupili vrijednosti koja je unesena u text areu. Isti je onome za pristupanje vrijednosti običnog text boxa. Znači, unesena

vrijednost se nalazi unutar superglobalne varijable $_POST ili $_GET, ovisno o metodi prosljeđivanja podataka forme. Pošto je u našem primjeru metoda POST vrijednost se nalazi unutar $_POST varijable.

U gornjoj slici je u text area polje unesena vrijednost sa slike ispod.

Znači radi se o tekstu od tri linije od kojih je jedna prazna. No, u našem ispisu je u prvom slučaju :

```
echo "<b>Unijeli ste : </b><br>" . $_POST["tekst"];
```

sve ispisano u jednoj liniji.

O čemu se radi?

Odgovor na ovo pitanje je zapravo vrlo očit. Radi se o tome da HTML u svom sourcu zanemaruje sve praznine, tj. više uzastopnih praznina interpretira poput jednog praznog prostora. Pod isto pravila spada i prelazak u novi red u HTML sourcu. Ukoliko želite, možete se u to uvjeriti ako pogledate source našeg primjera. U prvom ispisu unesene vrijednosti se HTML source nalazi u više redova, no izgled u browseru je kao da nema novih redova.

No tu nam dolazi u pomoć PHP sa svojim ugrađenim funkcijama. Ona je korištena u drugom ispisu u primjeru, i kao što vidite, ispis unesene vrijednosti prati isti format teksta koji je upisan u text area polje.

Korištena funkcija je :

```
nl2br($_POST["tekst"]);

(nl2br - čitaj new line to break)
```

Ime funkcije govori što ona radi. U stringu koji joj se da kao argument mijenja sve nove redove (\n) sa HTML
 tagom koji prebacuje ispis u novi red pri gledanju dokumenta u browseru.

Kod rada sa ovakvim tekstovima je potrebno paziti na neke stvari. Uzet ću za primjer nekakvu news skriptu. U njoj se vijesti unose kroz text area polje, pošto je to najjednostavniji način za unos većeg teksta. Te vijesti spremamo u neki trajni spremnik podataka, poput MySQL baze.

Očita stvar je da želimo pri ispisu vijesti zadržati format koji je zadao autor vijesti pri unosu. Znači kada stisne enter da se i na siteu vidi prijelaz u novi red.

Ovdje treba paziti kada će se pozvati funkcija nl2br. Naime, ukoliko se ta unesena vijesti spremi u bazu tako da se prije samog spremanja 'provuče' kroz nl2br te se ta vrijednost spremi u bazu, pri ponovnom uređivanju vijesti će se u text area polju vidjeti i
 tagovi, što nije poželjno. Stoga je preporučljivo funkciju nl2br pozivati netom prije ispisa vijesti na siteu, te u bazu spremiti izvornu vrijednost (onakvu kakva je unesena u text areu) da bi je mogli opet prikazati u text arei u njenom izvornom obliku.

Oko metoda uređivanja postojećih vrijednosti (npr. u slučaju naknadnog uređivanja vijesti) ćemo se više pozabaviti u slijedećem poglavlju članka. Za sada idemo dalje na elemente forme.
Hidden polje

```
<input type="hidden" name="nevidljivoPolje" value="Vrijednost polja">
```

Kao što i samo ime elementa / polja kaže, radi se o polju koje nije vidljivo pri gledanju dokumenta u browseru, te korisnik ne može direktno utjecati na vrijednost koja je spremljena u njega. Vrijednost koje se nalazi u u takvom polju se mora namijestiti direktno u sourcu dokumenta ili unosom u HTML

source ručno ili pomoću PHP-a. Drugi način promijene vrijednosti je dinamički, za vrijeme gledanje stranice u browseru putem JavaScripta.

Kao i kod text boxa, vrijednost se unosi u value atribut.

Također, pravila oko pristupa tim podacima su ista kao i kod text boxa, sa malenom razlikom. Ta je da u njega možemo spremiti tekst od više redova.

Poput ovoga :

```
<input type="hidden" name="nevidljivoPolje" value="Vrijednost polja

U više redova">
```

Ovo polje obično služi za pamćenje nekih vrijednosti koje ne ovise o korisniku (surferu), te ih na ovaj način štitimo od korisničkog unosa i izmjene. U primjeru 5 je već korišteno hidden polje za spremanje vrijednosti kontrolne varijable nav koja nam je služila za navigaciju po dokumentu kada smo promijenili metodu slanja iz POST u GET.
Checkbox

```
<input type="checkbox" name="jezik" value="hrvatski">
```

Checkbox je element koji se uglavnom koristi u slučaju da korisniku želimo omogućiti da izabere jednu / više / niti jednu vrijednost iz nekog logički povezanog skupa vrijednosti. Recimo, u gornjem primjeru bi značilo da naš korisnik / surfer priča hrvatski ukoliko je označio to polje, ili ga ne priča ukoliko ga je ostavio neoznačenim. Normalno, pošto se radi o grupi povezanih informacija trebali bismo ponuditi više jezika pomoću više checkboxova .

Iako gornja tvrdnja ne mora biti uvijek točna, recimo može se raditi o samo jednoj vrijednosti, poput označavanja suglasnosti sa nečim ili sličnim da / ne situacijama.

Kroz slijedećih par primjera ću ilustrirati oba načina korištenja ovog elementa.

Prvo idemo pogledati kako prihvatiti vrijednost checkboxa u PHP-u pa ćemo na tome graditi dalje. Ovaj primjer se može koristiti u situaciji u kojoj je potreba potvrda surfera da je punoljetan i da je suglasan sa pravilima našeg sitea. Recimo da imao neki web shop na kojem se prodaje alkoholno piće ovo bi bila realna situacija. Ali, siguran sam da se svi možete sjetiti barem još jednog tipa sitea gdje je ovo korišteno.

Primjer 7 a) : Prihvat checkbox elementa

```
<html>

<head>

<title>Primjer 7 a) : Prihvat checkbox elementa</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h2>Primjer 7 a): Prihvat checkbox elementa</h2>

<hr>
```

```
<?

if (! $_POST["SBgumb"] ){

?>

<form name="form1" method="post" action="">

<p>

<input type="checkbox" name="punoljetan" value="1">

Punoljetan sam i suglasan sam sa pravilima sitea</p>

<p>

<input name="SBgumb" type="submit" value="Ulaz">

</p>

</form>

<?

} else {

// provjera da li je označen checkbox

if ($_POST["punoljetan"]) {

echo "Pošto si punoljetan možeš dalje koristiti ove stranice";

} else {

echo "Pošto nisi punoljetan ne možeš pristupiti sadržaju ovih stranica";

die();

}

}

?>

</body>

</html>
```

Objašnjenje :

Mislim da je gornji primjer sam po sebi jasan. Način prihvata informacije iz checkboxa je identičan onome iz prijašnjih primjera.

Znači, unutar superglobalne varijable $_POST na indexu ekvivalentnom imenu checkbox elementa u formi se nalazi njegova vrijednost. Pošto se checkboxu obično zadaje predefinirana vrijednost u

samom HTML-u pomoću value atributa, ukoliko ga se označi u formi ta vrijednost će biti vidljiva PHP skripti. Ukoliko nije bio označen, njegova vrijednost neće biti vidljiva skripti.

Ovdje je potrebno paziti da se value atributu ne da vrijednost 0, jer u tom slučaju gornji primjer neće raditi. Ukoliko očekujete 0 kao vrijednost iza scheckboxa onda je potrebno zamijeniti

```
if ($_POST["punoljetan"]) {

sa

if (isset($_POST["punoljetan"])) {
```

što će provjeriti da li postoji bilo koja vrijednost u $_POST["punoljetan"].

Ovo je bio vrlo jednostavan primjer. Radilo se o da / ne situaciji. Idemo se sada vratiti na naš primjer sa jezicima. Recimo da želimo na našem siteu imati formu u kojoj ćemo ponuditi surferu par svjetskih jezika i pitati ga koji od njih tečno govori. Očito je da će neki od surfera označiti i više jezika, dok neki možda niti jedan. Ovo malo komplicira stvari.

Idemo to vidjeti na primjeru.

Primjer 7 b) : Prihvat vrijednosti više povezanih checkbox elemenata

```
<html>

<head>

<title>Primjer 7 b) : Prihvat vrijednosti više povezanih checkbox elemenata</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

</head>

<body>

<h2>Primjer 7 b) : Prihvat vrijednosti više povezanih checkbox elemenata</h2>

<hr>

<?

if (! $_POST["SBgumb"] ){

?>

<form name="form1" method="post" action="">

<p>Koje od ponuđenih jezika pričate tečno?</p>

<p>

<input type="checkbox" name="jezik[]" value="engleski">

Engleski<br>
```

```
<input type="checkbox" name="jezik[]" value="njemački">

Njemački<br>

<input type="checkbox" name="jezik[]" value="francuski">

Francuski<br>

<input type="checkbox" name="jezik[]" value="ruski">

Ruski </p>

<p>

<input name="SBgumb" type="submit" id="SBgumb" value="Pošalji">

</p>

</form>

<?

} else {

if (is_array($_POST["jezik"]) and count ($_POST["jezik"]) > 0 ){

echo "Jezici koje pričate su :";

foreach ($_POST["jezik"] as $pricam){

echo $pricam . ",";

}

} else {

echo "Jeste sigurni da ne pričate niti jedan od navedenih jezika?";

}

}

?>

</body>

</html>
```

Objašnjenje :

Kao što naziv primjera kaže, radi se o formi koja sadrži više povezanih checkbox elemenata. Njihovo povezivanje u jednu cjelinu smo obavili tako da smo svima dali isto ime. U ovom slučaju name="jezik[]". No to nije sve što smo napravili. Primjećujete uglate zagrade na kraju imena?

Čemu one služe?

Odgovor na to pitanje leži duboko unutar samog PHP-a i njegovog načina baratanja sa podacima koji dolaze iz forme.