



IT255 - VEB SISTEMI 1

Serverski skriptovi – PHP jezik

Lekcija 07

PRIRUČNIK ZA STUDENTE

IT255 - VEB SISTEMI 1

Lekcija 07

SERVERSKI SKRIPTOVI – PHP JEZIK

- ✓ Serverski skriptovi – PHP jezik
- ✓ Poglavlje 1: Instalacija i priprema za rad
- ✓ Poglavlje 2: Osnove programiranja u PHP-u
- ✓ Poglavlje 3: Nizovi i matrice
- ✓ Poglavlje 4: Konstante i operatori
- ✓ Poglavlje 5: Upravljačke strukture u PHP-u
- ✓ Poglavlje 6: Upotreba PHP-a - izrada formi
- ✓ Poglavlje 7: Izvršno PHP okruženje na strani servera
- ✓ Poglavlje 8: PHP i baze podataka
- ✓ Poglavlje 9: Pokazna Vežba 7
- ✓ Poglavlje 10: Domaći zadatak 7
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

Neki operativni sistemi, kao što su Linux i mnog everzije Unix-a, sada se isporučuju sa već instaliranim PHP-om. Kod drugih operativnih sistema, kao što su Windows ili Mac OSX, ne.

PHP je skript programski jezik opšte namene. Prvobitno je dizajniran kao jezik za kreiranje dinamičkih veb strana. Omogućava brzo procesiranje i učitavanje strana, jednostavan je za razumevanje i korišćenje i izvršava se na skoro svim operativnim sistemima. PHP se procesira od strane veb servera i generiše XHTML kod ili neki drugi izlaz koji veb čitačimogu da prepoznaju i interpretiraju.

Da biste mogli da kreirate interaktivne vebstrane i da izvršavate PHP programe, neophodno je da imate pristupnekom serveru koji podržava PHP. Međutim, često je u toku kreiranja nekog interaktivnog veb sajta pomoću PHP-a veoma nepraktično da svaki put kada napravite neku PHP stranu, morate da je prebacite na udaljeni server kako biste je istestirali. Zbog toga se preporučuje da PHP instalirate lokalno, na svoj računar.

U prvim danima Veba uloga servera bila je da sa spoljne memorije učitava traženi statični sadržaj (HTML dokument, sliku i drugo) i da ga u okviru HTTP protokola pošalje nazad klijentu. Dinamičko kreiranje sadržaja, sa druge strane, pristup je kod koga klijent serveru šalje standardan zahtev za određenim sadržajem, a taj zahtev se obrađuje odgovarajućom programskom procedurom na serveru.

Sintaksa programskog jezika PHP je slična jezicima kao što su C, C++, Java i Perl. U odmah primetne specifičnosti PHP-a spadaju fleksibilni tipovi promenljivih, odnosno mogućnost promene tipa tokom izvršavanja i eliminisanje potrebe da se on definiše unapred, označavanje promenljivih znakom za dolar, mogućnost i proceduralnog i objektno orijentisanog programiranja, i nedostatak inicijalne funkcije/kalse.

Strukture grananja omogućavaju da se u zavisnosti od trenutne vrednosti ili stanja određenih promenljivih i konstanti izvrše različiti delovi programa. PHP programski jezik podržava dve osnovne strukture grananja.

Sintaksa PHP-a je određena sa PHP tagom, koji pokazuje serveru gde počinje, a gde se završava PHP kod u okviru HTML koda.

▼ Poglavlje 1

Instalacija i priprema za rad

OSNOVNO O PHP JEZIKU

PHP je skript programski jezik opšte namene. Prvobitno je dizajniran kao jezik za kreiranje dinamičkih veb strana.

PHP je u velikoj meri u upotrebi danas u Internet programiranju (uključujući i neke od najpopularnijih dinamičkih web sajtova), postoji veliki broj ne-trivijalnih primera i besplatnih resursa.

Jezik je konvencionalan, uključuje mnoge moćne i korisne mogućnosti, podržava više različitih programerskih stilova.

Aktuelan jezik, dizajniran da se jednostavno prilagodi najsavremenijim dizajnima programskih jezika.

PHP je skript programski jezik opšte namene. Prvobitno je dizajniran kao jezik za kreiranje dinamičkih veb strana. Omogućava brzo procesiranje i učitavanje strana, jednostavan je za razumevanje i korišćenje i izvršava se na skoro svim operativnim sistemima. PHP se procesira od strane veb servera i generiše XHTML kod ili neki drugi izlaz koji veb čitačimogu da prepoznaju i interpretiraju.

Malo istorije

- Rasmus Lerdorf
1994 – Personal Home Page Tools
PHP (PHP: Hypertext Preprocessor)
PHP2
Integracija sa bazom podataka
- PHP3
Novi parser
- PHP4
Zend engine
<http://www.zend.com/>
- PHP5
realizovan Aprila 2004

ISTORIJA PHP JEZIKA - NOVE VERZIJE

PHP je skript jezik na serverskoj strani Web aplikacije koji se integriše u okviru HTML dokumenata.

PHP 5.6+ i PHP 7+

U izdanje PHP 7 okruženja dodata su odlična poboljšanja. U verziji PHP 5.5+ je dramatično poboljšana bezbednost. U ovoj lekciji ukazujemo da treba upotrebiti najnoviju PHP alatku za enkripciju „password hash“, umesto alatke MD5, koja se koristi u mnogim aktuelnim knjigama. U poslednjih nekoliko godina alatka MD5 se pokazala kao „ranjiva“ na napade hakera.

„PHP 7 je zasnovan na PHPNG projektu (PHP Next-Gen), koji vodi Zend za ubrzanje PHP aplikacija. Poboljšanje performansi ostvareno iz PHP-a 7 je ogromno i varira između 25 i 70 odsto od aplikacija u realnom svetu, i sve to samo nadgradnjom PHP-a, bez potrebe da menjate i jednu liniju koda!“ — www.zend.com .

PHP 7 takođe zamenjuje fatalne greške, koje su prethodno zaustavljale program, sa izuzecima kojima može da se rukuje unutar samog programa.

Ako prelazite sa prethodne verzije PHP-a na PHP 7, pogledajte sledeći link

<http://php.net/manual/en/migration70.php>

Kod koji je upotrebljen u primerima u ovoj lekciji je kompatibilan sa verzijom PHP 7. Većina primera je, takođe, kompatibilna sa verzijama PHP 5.5 i PHP 5.6.

PHP je dizajniran tako da pomaže programerima da razvijaju dinamičke i podacima vođene Web stranice.

PHP na jednostavan način pristupa MySQL, kao i drugim open source bazama podataka, kao što je PostgreSQL.

MySQL mora da bude instaliran zajedno sa funkcionalnim Web serverom, ali to je jednostavan korak u podešavanju PHP okruženja.

PHP je skript jezik na serverskoj strani Web aplikacije koji se integriše u okviru HTML dokumenata.

Takođe, moguće je ubacivati HTMLkod u PHP skript. PHP skript se parsira i interpretira na serverskoj strani Web aplikacije.

PHP budućnost je obećavajuća, jer je popularan kod Web programera i dizajnera, a moćan je i jednostavan za korišćenje.

OSNOVNO O PHP-U NA RAČUNARU

Da biste mogli da kreirate interaktivne vebstrane i da izvršavate PHP programe, neophodno je da imate pristupnekom serveru koji podržava PHP.

Može se izvršavati bez Web servera, ali:

-> Ni malo zabavno

-> Ni malo praktično

Mi ćemo koristiti Apache, PHP, MySQL kombinaciju jer:

-> Veliki broj aplikacija se izvršava na ovoj platformi

-> Odlična Open Source kombinacija

Neki operativni sistemi, kao što su Linux i mnog everzije Unix-a, sada se isporučuju sa već instalanim PHP-om. Kod drugih operativnih sistema, kao što su Windows ili Mac OSX, morate to da uradite sami.

Dakle, PHP je skript jezik čija je primarna, ali ne i jedina funkcija (jer je u pitanju jezik opšte namene) obrada podataka na veb serveru i njihova implementacija u HTML kod.

Autor je jezik kreirao radi održavanja vlastitih web stranica, a na bazi tada jako zastupljenog Perl jezika. Inače, sintaksa samog jezika je vrlo slična sintaksama jezika C i Perl. Ipak, potrebno je (osim, naravno, poznavanja samog jezika), da bi se uspostavila funkcionalnost, ispuniti nekoliko uslova:

- *Instalirati PHP na računaru na kome želimo da se izvršava PHP skripte se interpretiraju, što znači da se ne startuju direktno već putem programa za interpretaciju.* Instalacijom PHP paketa na računar, ova komponenta biće omogućena.
- *Instalirati veb server na računaru na kome želimo da se izvršava PHP,* zapravo i nije potreban veb server da bise izvršavao, već samo interpreter. Ali, obzirom da ćemo se držati PHP-a u kontekstu veba, neophodan je i veb server.
- *Instalirati bazu podataka* na računaru na kome želimo da se izvršava PHP ima dosta built-in konekcija koje dozvoljavaju interakciju sa bilo kojom back-end bazom podataka, gde Web aplikacija smešta podataka. Prvi deo Web aplikacija, još se naziva i Prezentacioni nivo, je ono što korisnik vidi. MySQL je najčešće korišćenaback-end baza podataka za organizaciju Web sadržaja zato što je besplatna (*open source*), podržava veliki broj korisnika i platformi, i poseduje moćan i jednostavan SQL interfejs. U velikom broju slučajeva ne koristi se samo PHP. Uobičajeno je da je potrebna baza podataka koja sadrži informacije za dinamički Web sadržaj.

INSTALACIJA PHP-A

Takođe, MySql server/baza podataka ne moraju biti na istom računaru na kome je i veb server. Veb server ne mora biti samo na jednom računaru.

- *Odabrali alat za PHP kodiranje Izvorni PHP kod je moguće pisati u najprimitivnijim alatima za procesiranje teksta (npr. Notepad).* Ipak, najbolje je koristiti neki alat koji prepoznaje (i markira) PHP naredbe. Moguće je (i poželjno) u ovu svrhu koristiti i neki od (najčešće komercijalnih) paketa za proizvodnju HTML strana (Adobe Dreamweaver, MS Front Page itd.) koji u sebi ima i deo za dizajniranje.

Instalacija PHP-a u operativni sistem

Postoji više načina da se PHP instalira na operativni sistem. Ukoliko je u pitanju Linux, instalacija se vrši pri ili posle instalacije samog OS-a pridruživanjem odgovarajućih servisa iz distribucije OS-a.

Kada je u pitanju *MS Windows*, instalacija se generalno može izvršiti na dva načina. Jedan je „ručno“, komponentu po komponentu, a drugi instalacijom nekog paketa koji u sebi sadrži ceo sistem. Ovakvi paketi su izuzetno praktični, i mi ćemo se orijentisati na jedanodnjih). Njime ćemo pokriti tri gore pomenuta uslova (PHP, veb server i bazu).

Za potrebe ovih predavanja, preuzmite i instalirajte WampServer sa Interneta (<http://www.wampserver.com/en/>).

PHP na strani *web servera* predstavlja *pretprocesor* kome se prosleđuju PHP skripte.

Ovo u praksi radi na sledeći način: kreirate HTML stranice i u njih dodate i svoje PHP skripte, date stranice obavezno imaju ekstenziju "**.php*". Kada ih postavite na *web server* i korisnik ih zatraži putem svog *browser*-a, web server će na osnovu ekstenzije prepoznati da se radi o PHP stranicama i prosledi će ih instaliranom PHP pretprocesoru. Zatim će *pretprocesor* izvršiti programski kod i rezultat vratiti web serveru, koji nakon toga sve šalje *browser*-u.

Rezultat procesiranja su najčešće dinamički kreirane HTML stranice, koje se zasnivaju na podacima iz neke od baza podataka, najčešće MySQL.

U slučaju da i sami održavate neki *veb sajt*, sigurno ste se susreli sa problemom održavanja sajta i to kada broj stranica pređe kritičan broj.

U tom slučaju, kada imamo veliki broj stranica, svako ažuriranje (*update*) je vrlo mukotrpan. U tom momentu bi trebalo da se okrenete PHP-u ili ASP-u, a sve u zavisnosti od toga na kojoj platformi radi vaš host provajder.

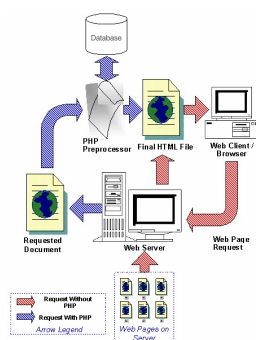
INSTALACIJA PHP - NASTAVAK

Uloga svakog veb servera svodi se, manje ili više, na jednu funkciju, a to je prihvatanje određenog zahteva od korisnika putem Interneta, obrada tog zahteva i vraćanje rezultata obrade

Ono što je bitno, to je da u radnom okruženju stvorimo uslove bliske produkcijom, kako bi implementacija prošla što bezbolnije.

Shema funkcionisanja PHP-a

PHP je program koji izvršava dinamičke HTML skriptove. Interpreter jezika direktno izvršava komande koje generišu izlaz u formi HTML/HTTP (bolje nego korišćenje kompajlera koji kao rezultat prevodi komande jezika u mašinski kod). Server zna da izvrši PHP kod kada zahtevani fajl ima *.php* ekstenziju. (Slika 1). Ukoliko korisnički zahtev sadrži naziv i tip dokumenta koji postoji na veb serveru u nekom statičkom obliku, veb server će jednostavno preuzeti dokument sa fajl sistema i isporučiti ga Internetom korisniku.



Slika 1.1 Shema funkcionisanja PHP-a

Ali, ukoliko zahtev ne podrazumeva već postojeći dokument, već zahteva sadržaj koji će biti dinamički generisan, veb server mora stvoriti odgovor tako što će startovati skriptu koju zahtev sadrži, te njen rezultat implementiran u dokument u kome se skripta nalazi i proslediti korisniku. Na kraju, obzirom da je pomenuta skripta/program implementirana u neki dokument, korisnik će dobiti svoj rezultat u formi validnog HTML dokumenta.

Prilikom ovog procesa, veb server i klijent (a kada kažemo klijent mislimo na pretraživač IE, FireFox...) koriste isti „jezik” protokol za komunikaciju. Taj „jezik” naziva se HTTP (HyperText Transfer Protocol) i osnovni je način komunikacije između računara na Internetu.

Gde se u svemu tome uklapa PHP? Uloga PHP-a je u stvari da prevede i startuje pomenutu skriptu, te da prosledi dobijeni rezultat klijentu. Treba znati da PHP nije jedini jezik koji može uraditi tako nešto. Postoje i drugi jezici koji se dobro snalaze u ovom „poslu” (C#, VB, Java...), uz preduslov da server zna koji od njih i kada treba da startuje i gde da pronade interpretere za njih.

IDENTIFIKACIJA SKRIPT JEZIKA

Dakle, onog trenutka, kada je PHP izvršio skriptu, generisao HTML uz pomoć te skripte i prosledio stranu, toj strani više ne može pristupiti.

Da bi veb server prepoznao zahtev korisnika za izvršenjem neke skripte potrebno je da bude ispunjen najmanje jedan uslov. Taj uslov je da dokument ima odgovarajuću ekstenziju. U slučaju PHP-a, ova ekstenzija je .php. Da je u pitanju neki drugi jezik, ekstenzija bi bila drugačija (npr. C#, VB, J#) .aspx ili .asp, java-.jsp itd.)

Drugi uslov je da dokument uopšte sadrži skriptu u sebi. Ovo nije uistinu preduslov da se dokument prosledi korisniku, jer će, čak i ukoliko ne sadrži skriptu, njegov sadržaj biti prosleđen. Jednostavno, veb server će startovati PHP (interpreter) koji će proći kroz dokument ali obzirom da u njemu nema skripte, ništa neće biti startovano pa će takav dokument, u svom izvornom obliku, biti prosleđen klijentu. Ali, ukoliko želimo (a želimo) da se neka skripta u našem dokumentu izvrši, moramo to dati do znanja interpreteru. Odgovor leži u parsiranju.

Znamo da se ceo HTML „svet” sastoji od tagova, odnosno, markera koji ograničavaju i identifikuju određene celine. Kada se strana sa ekstenzijom .php prosledi PHP-u, biva parsirana tako što se analiziraju svi njeni tagovi. PHP sve „nezanimljive” tagove prosleđuje

u formi u kojoj ih je i pročitao sve dok ne dođe do taga koji se tiče samo njega. Taj tag se obeležava sa `<?php ?>` ili `<? ?>`.

Kada PHP parser prepozna anavedeni tag, dolazi do aktivacije sadržaja unutar taga i skripta biva izvršena.

Treba obratiti pažnju na to da smo PHP tagove označili sa `<?php ?>` ili `<? ?>` i ako ovo ili << ne mora u svakom slučaju biti pravilo. Naime, podrazumevano setovanje konfiguracije PHP-a je da prepoznaje samo jasno naznačene, »dugačke« otvarajuće tagove. Ovo setovanje najčešće biva promenjeno tako da PHP prepoznaje i skraćene tagove, ali za slučaj da su setovanja na produkcionom serveru (onome na kome želimo da trajno držimo naš sajt) ostala na podrazumevanim vrednostima, dugački otvarajući tag je sigurna varijanta, jer najverovatnije, nećemo (ukoliko setovanja ne promenimo na nivou skripte) imati mogućnost pristupa konfiguracionim fajlovima za slučaj da je ova opcija isključena. U sklopu kursa, koristićemo isključivo dugačke otvarajuće tagove.

Veoma je bitno (iako je logično iz prethodnog teksta) naglasiti, da PHP ne omogućava nikakvu aktivnost nakon što je dokument prosleđen klijentu. Dakle, onog trenutka, kada je PHP izvršio skriptu, generisao HTML uz pomoć te skripte i prosledio stranu, toj strani više ne može pristupiti. Zbog toga u PHP-u ne možemo napraviti animacije ili bilo kakve pokrete u pretraživaču. Zato koristimo druge programske alate, poput JavaScript-a, ActionScript-a ili Jave.

▼ Poglavlje 2

Osnove programiranja u PHP-u

VAŠ PRVI PHP SKRIPT

Svaka PHP skripta počinje sa "<?php", a završava se sa ">?". Skripte se ugnežđavaju u standardne HTML stranice, ali mogu biti i u eksternim fajlovima.

Ako ste pratili prethodni objekat učenja i podesili sve onako kako je opisano, onda ste spremni da krenete u kodiranje. Pošto je instaliran i *web server* i *podrška za PHP*, pre samog početka potrebno je da pokrenete *Apache* server.

Za kreiranje *PHP* skripti potreban vam je bilo kakav tekst editor. To može da bude i onaj koji dolazi uz *Windows* - "*Notepad*", ali za bilo kakav komotniji rad preporučujemo dva druga editora, a to su:

1. *Note Tab Light* - koji možete preuzeti sa adrese www.notetab.com
2. *HTML Beauty* - koji možete preuzeti sa adrese www.beauty.com

Kodiranje skripti ćemo započeti od programerskog standarda, programa "*Zdravo svete!*" (tj. "*Hello World!*") iz koga ćemo videti samu sintaksu *PHP*-a. Otvorite neki od editora i ubacite sledeći kod:

```
<html>
<head>
<title>PHP pocetak</title>
</head>
<body>
<?php echo "Zdravo svete!"; ?>
</body>
</html>
```

Snimite fajl, obavezno sa ekstenzijom "**.php*" ili "**.php3*", "**.phtml*". Preporuka je da to bude "*.php*", jer je to deo *PHP* standarda. Mada, fajl možete snimiti i sa ekstenzijom *.htm* ili *.html*, s tim što tada morate da podesite server da i takve fajlove, pre slanja u browser, propusti kroz *PHP parser* (pretprocesor). Fajl, za početak, obavezno snimite u folder "*C:\nusphere\apache\Htdocs*", koji je *root* folder vašeg servera. Sada ćemo isprobati naš mali *PHP* skript, a ujedno ćemo i istestirati server.

Otvorite *browser* i u *Address* liniju ukucajte " <http://localhost/proba.php>". Dobićete ekran kao na slici, znači samo jednu rečenicu. Izvorni kod ćete videti preko menija " *View > Source*", a rezultat parsiranja će biti:

```
<HTML>
<HEAD>
```

```
<TITLE>PHP pocetak</TITLE>
</HEAD>
<BODY>
Zdravo svete!
</BODY>
</HTML>
```

Kao što ste i videli u primeru, svaka **PHP** skripta počinje sa "`<?php`", a završava se sa "`?>`". Skripte se ugnežđavaju u standardne HTML stranice, ali mogu biti i u eksternim fajlovima. Ovde smo za ispisivanje teksta u *browser*-u iskoristili funkciju "`echo`", a cilj skripta je osnovno upoznavanje sa **PHP** sintaksom. Ovo je bilo dovoljno za početak, nastavak sledi.

SINTAKSA PROGRAMSKOG JEZIKA

Sintaksa programskog jezika PHP je slična jezicima kao što su C, C++, Java i Perl.

Sintaksa programskog jezika PHP je slična jezicima kao što su C, C++, Java i Perl. U odmah primetne specifičnosti PHP-a spadaju fleksibilni tipovi promenljivih, odnosno mogućnost promene tipa tokom izvršavanja i eliminisanje potrebe da se on definiše unapred, označavanje promenljivih znakom za dolar, mogućnost i proceduralnog i objektno orijentisanog programiranja, i nedostatak inicijalne funkcije/kласe.

Promenljive i konstante

U osnovne tipove promenljivih u PHP programskom jeziku spadaju:

- logička vrednost (engl. `boolean`),
- celobrojna (engl. `integer`) i decimalna (engl. `float`) vrednost i
- niz simbola (engl. `string`)

U složene tipove promenljivih u PHP programskom jeziku spadaju:

- nizovi elemenata (engl. `array`) i
- objekti kao instance klasa (engl. `object`)

Poseban tip promenljivih u PHP programskom jeziku su tzv. resursi

(engl. `resource`) u kojima se čuvaju uspostavljene veze ka bazi podataka, fajlovima i slično. Dodatno, promenljiva ne mora imati vrednost i tip ukoliko je u „nultom“ stanju (engl. `null`).

Tip i vrednost promenljive se tokom izvršavanja mogu više puta menjati:

```
x = 1; x = 2.2; x = "TeKCT"; x = array(1,2,3,4); x = new stdClass(); x = null;
```

Pri dodeljivanju vrednosti promenljivoj može se eksplicitno zadati kog tipa ona treba da bude :

```
$x = 142.857;
```

```
y = (int) x;
```

Pri tom, rezultat izvršavanja navedenog skripta će promenljivoj u dodeliti vrednost 142, odnosno početni deo zadate vrednosti koji se mogao interpretirati kao ceo broj, dok je za zaokruživanje potrebno koristiti namensku funkciju.

VIDLJIVOST PROMENLJIVIH

Pod vidljivošću promenljivih podrazumeva se njihova oblast važenja, odnosno oblast programa u kojoj se promenljiva može koristiti.

Usled fleksibilnosti rada sa promenljivima u PHP-u se mogu upoređivati promenljive različitih tipova. Pri tom se za tzv. „meko“ poređenje, koje uzima u obzir samo vrednost, koristi operator „==“, dok se za tzv. „tvrdo“ poređenje, koje u obzir uzima i tip promenljivih, koristi operator „===“:

```
x = (int)123; y = (float) 123;if ( x == y) echo 'Ista vrednost'; // Daif ( x === y) echo 'Ista vrednost i tip '; // Ne
```

Da bi se pristupilo određenoj promenljivoj njen naziv se može koristiti kao vrednost druge promenljive (tzv. promenljive promenljive):

```
$ime = „Petar“;  
$pp = "ime";  
echo $$pp; // Ispisuje „Petar“
```

Pri davanju naziva promenljivama treba biti oprezan kod korišćenja znaka „_“ kao početka imena jer njime počinju i nazivi sistemskih, predefinisanih promenljivih kao što su `$_GET`,

`$_POST`, `$_REQUEST`, `$_SERVER`, `$_SESSION` i slične.

Nasuprot promenljivama, tip i vrednost jednom definisane konstante ostaju isti do kraja izvršavanja programa, a za definisanje konstanti koristi se funkcija *define*:

```
define("PI", 3.14);
```

U nazivima promenljivih i konstanti pravi se razlik između velikih i malih slova.

Vidljivost promenljivih

Pod vidljivošću promenljivih podrazumeva se njihova oblast važenja, odnosno oblast programa u kojoj se promenljiva može koristiti. Oblast važenja promenljive direktno zavisi od mesta na kojem je promenljiva inicijalno deklarirana. Na primer, promenljiva koja je definisana van bilo koje funkcije smatra se globalnom promenljivom:

```
<?php $x = 3;
```

i njoj se može pristupiti iz globalnog prostora, odnosno van funkcija.

UPOTREBA NAVODNIKA

U php-u se tekstualni podaci - nizovi znakova (engl. string) - definišu unutar jednostrukih ili dvostrukih znakova navoda.

Sa druge strane, promenljive deklarisanе u okviru funkcija se nazivaju lokalnim promenljivama i mogu se koristiti samo unutar te funkcije:

```
<?php $x = 3; function IspisBroja(){$x = 4; echo $x; IspisBroja(); // Ispisa $x; // Ispisaće
    celokalnu vrednost 4 echo
```

lokalnu vrednost 3

Ukoliko u nekoj funkciji želimo da koristimo globalnu promenljivu, u tu svrhu ćemo iskoristiti reč global:

```
<?php $x = 3; function IspisBroja() {global $x; echo $x; } IspisBroja(); // Ispisaće lokalnu vrednost
3
```

Sistemske, predefinisane promenljive kao što su `$_COOKIE`, `$_ENV`, `$_FILES`, `$_GET`, `$_POST`, `$_REQUEST`, `$_SERVER` i `$_SESSION` se smatraju superglobalnim i mogu se koristiti u bilo kojoj oblasti programa.

Upotreba navodnika

U php-u se tekstualni podaci - nizovi znakova (engl. string) - definišu unutar jednostrukih ili dvostrukih znakova navoda. Upotreba jednostrukih i dvostrukih znakova navoda je gotovo identična, ali određene razlike ipak postoje. Glavna razlika se nalazi u upotrebi promenljivih unutar znakova navoda. Na primer, kod:

```
$procenat = 80;
```

```
echo 'PHP se koristi na preko $procenat procenata sajtova';
```

ispisaće:

PHP se koristi na preko \$procenat procenata sajtova

UPOTREBA NAVODNIKA - NASTAVAK

Dakle, tekst unutar jednostrukih navodnika se ne interpretira, dok se u tekstu unutar dvostrukih navodnika pronalaze oznake promenljivih i na njihova mesta se postavljaju odgovarajuće vredno

Dok će kod:

```
$procenat = 80;
```

```
echo " PHP se koristi na preko $procenat procenata sajtova";
```

ispisati:

PHP se koristi na preko 80 procenata sajtova . Dakle, tekst unutar jednostrukih navodnika se ne interpretira, dok se u tekstu unutar dvostrukih navodnika pronalaze oznake promenljivih i na njihova mesta se postavljaju odgovarajuće vrednosti. To dalje znači da će se kod sa jednostrukim navodnicima brže interpretirati od ekvivalentnog koda koji koristi dvostruke navodnike.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 3

Nizovi i matrice

NIZOVI SA NUMERIČKIM INDEKSIMA

PHP podržava numerički i asocijativno indeksovane jednodimenzionalne i višedimenzionalne nizove

PHP podržava numerički i asocijativno indeksovane jednodimenzionalne i višedimenzionalne nizove, odnosno matrice. To znači da se kao indeks može koristiti redni broj elementa ili simbolički niz karaktera, kao i da broj indeksa nije ograničen. Tip vrednosti elemenata niza nije ograničen i može se razlikovati za različite članove. Za deklarisanje novog niza koristi se funkcija `array`.

Niz (`array`) je promenljiva koja sadrži skup vrednosti u nekom redosledu. Jedan niz može imati više elemenata, a svaki element sadrži jednu vrednost (na pr. tekst, broj, drugi niz). Svakom elementu niza pridružen je indeks pomoću koga se pristupa elementu.

PHP podržava:

- numerički indeksirane nizove, koji su slični nizovima u drugim programskim jezicima
- asocijativne nizove, kod kojih indeksi mogu biti reči ili druge informacije

U PHP-u, numerički indeksi počinju nulom, mada se umesto ove može zadati i neka druga vrednost.

Inicijalizacija niza sa numeričkim indeksima se obavlja primenom jezičke konstrukcije `array()`.

```
$proizvodi = array('Olovka', 'Sveska', 'Knjiga');
```

Niz se ne mora inicijalizovati ručno:

- Ako se podaci već nalaze u drugom nizu, mogu se iskopirati u dati niz primenom operatora `=`.
- Ako se želi niz rastućih brojeva, može se upotrebiti funkcija `range()`.

```
$brojevi = range(1, 10);
```

```
$neparni = range(1, 10, 2);
```

```
$slova = range('a', 'z');
```

Pristupanje sadržaju niza se obavlja pomoću imena niza i indeksa koji određuje element kome se pristupa. Indeks se zadaje u okviru uglastih zagrada.

```
$proizvodi[0]
```

Sadržaj elementa niza menja se pomoću operatora =.

```
$proizvodi[0] = 'Lenjir';
```

ASOCIJATIVNI NIZOVI

Sadržaj niza se može prikazati i posebnom, foreach petljom za rad sa nizovima.

Dodavanje novog elementa na kraj niza se obavlja na sledeći način:

```
$proizvodi[3] = 'Bojice';
```

Sadržaj niza se može prikazati sledećim kodom:

```
echo " proizvodi[0] proizvodi[1] proizvodi[2] proizvodi[3]"; ili u for petlji for ( i = 0; i < 3; i++) echo " proizvodi[" . $i . "];
```

i posebnom, **foreach** petljom za rad sa nizovima (svaki element niza se redom smešta u promenljivu \$tekuci čija se vrednost onda ispisuje):

```
foreach ( $proizvodi as $tekuci)
```

```
echo "$tekuci. ";
```

Asocijativni nizovi

U PHP-u postoje nizovi u kojima se svakom elementu može pridružiti indeks koji želimo.

Inicijalizacija niza u kome su imena proizvoda indeksi, a cene vrednosti:

```
$cene = array('Olovka'=>10, 'Sveska'=>30, 'Knjiga'=>150);
```

Isti niz se može inicijalizovati i samo jednim elementom, uz dodavanje ostalih:

```
$cene = array('Olovka' => 10); $cene['Sveska'] = 30; $cene['Knjiga'] = 150;
```

ili bez eksplicitnog pravljenja niza:

```
$cene['Olovka'] = 10;
```

```
$cene['Sveska'] = 30;
```

```
$cene['Knjiga'] = 150;
```

Sadržaju navedenog niza se može pristupiti u obliku:

```
$cene['Olovka'], $cene['Sveska'], $cene['Knjiga']
```

Rad u petlji sa asocijativnom nizovima zahteva korišćenje foreach petlje, ili funkcija list() i each().

RAD U PETLJI SA ASOCIJATIVNOM NIZOVIMA

PHP podržava formiranje M-dimenzionalnih matrica, odnosno matrica kod kojih je broj dimenzija ograničen samo arhitekturom i memorijom računara na kome se program izvršava.

Struktura foreach petlje u ovom slučaju uključuje i indekse:

```
foreach ( cene as indeks => $vrednost );
echo indeks. '=>'. vrednost. ';
```

Sadržaj niza se može nabrajati pomoću funkcije each():

```
while ( element = each( cene ) ) {
echo $element['key'];
echo ' - ';
echo $element['value'];
echo ' ';}
```

Funkcija each() čita tekući element niza i pomera interni pokazivač na naredni element. Pošto se poziva unutar *while* petlje, ona redom poziva elemente niza i zaustavlja se kad dođe do njegovog kraja.

U navedenom kodu, promenljiva \$element je niz od dva elementa.

Sadržaj niza se može nabrajati i pomoću funkcije list(). Dve vrednosti koje daje funkcija each() mogu se razdvojiti na sledeći način:

```
list( proizvod, cena ) = each( $cene );
```

Funkcija list pretvara elemente 0 i 1 iz niza koji je vratila funkcija each u dve nove promenljive \$proizvod i \$cena. Ceo niz \$cene može se obraditi u petlji:

```
while ( list( proizvod, cena ) = each( cene ) ) echo " proizvod - $cena";
```

PHP podržava formiranje M-dimenzionalnih matrica, odnosno matrica kod kojih je broj dimenzija ograničen samo arhitekturom i memorijom računara na kome se program izvršava:

```
$matrica = array(); for ($i=0; $i<3; $i++) for ($j=0; $j<3; $j++) for ($k=0; $k<3; $k++) for ($l=0; $l<3; $l++)
$matrica[$i][$j][$k][$l] = "$i-$j-$k-$l";
```

U višedimenzionalnim matricama, kao i u jednodimenzionalnim, tip indeksa i vrednosti, kao i broj elemenata u redovima, može se razlikovati. Pri tom na nizove i matrice u PHP programskom jeziku ne treba gledati kao na složene i apstraktne geometrijske strukture podataka, već pre kao na jednostavan sistem za njihovo organizovanje i adresovanje.

▼ Poglavlje 4

Konstante i operatori

KONSTANTA U PHP-U

Konstanta predstavlja vrednost koja se jednom zadaje i više se ne može menjati.

Konstanta predstavlja vrednost koja se jednom zadaje i više se ne može menjati:

- Da bi se olakšalo razlikovanje promenljivih i konstanti, usvojeno je da se imena konstanti pišu velikim slovima (mada nije obavezno).
- Ispred konstante se ne piše \$.
- Ako se želi upotrebiti vrednost konstante, navodi se samo njeno ime.
- Konstante mogu sadržati samo podatke skalarnog tipa, tj. logičkog tipa, celobrojnog, zakovnog i numeričke podatke s pokretnim zarezom.

U izvršnom PHP okruženju definisan je veliki broj konstanti, koje se mogu pregledati pomoću komande `phpinfo()`.

Primer: Cene artikala se mogu definisati kao konstante.

```
define('OLOVKACENA', 20);define('SVESKACENA', 45);
```

Operatori

Operatori su simboli koji omogućavaju izvršavanje operacija nad vrednostima i promenljivama.

U PHP-u postoje sledeće vrste operatora:

- Aritmetički operatori
- String operator (operator za konkatenciju, tj. spajanje stringova)
- Operatori dodele (na pr. `=`, `+=`, `-=`, i dr.)
- Operatori poređenja
- Logički operatori
- Operatori nad bitovima
- Operatori za rad sa nizovima (na pr. `[]` za pristup elementu niza)
- Ostali operatori (na pr. operator zanemarivanja greške, uslovni operator, i dr.)

Aritmetički operatori seobično primenjuju na brojeve. Ako se primene na string, pokušaće se njegovo pretvaranje u broj. Ako string sadrži *e* ili *E*, pretvoriće se u float, inače u ceo broj kao što je u sledećoj tabeli na slici 1 prikazano.

Aritmetički operator	Ime	Primer
+	sabiranje	\$a + \$b
-	oduzimanje	\$a - \$b
*	množenje	\$a * \$b
/	deljenje	\$a / \$b
%	moduo	\$a % \$b

Slika 4.1 Aritmetički operatori

ARITMETIČKI I OPERATORI POREĐENJA U PHP-U

Izrazi u kojima se koriste operatori poređenja vraćaju logičku vrednost true ili false.

Izrazi u kojima se koriste operatori poređenja vraćaju logičku vrednost *true* ili *false* su dati u tabeli na slici 2.

Operator `==` se često meša sa operatorom dodele `=`, što ne vodi željenom rezultatu.

Logički i operatori nad bitovima su prikazani u tabelama na slici 3, a operatori za rad sa nizovima su dati na slici 4.

Operatori poređenja	Ime	Primer
<code>==</code>	jednako	<code>\$a == \$b</code>
<code>!=</code>	različito	<code>\$a != \$b</code>
<code><</code>	manje od	<code>\$a < \$b</code>
<code>></code>	veće od	<code>\$a > \$b</code>
<code>>=</code>	veće ili jednako	<code>\$a >= \$b</code>
<code><=</code>	manje ili jednako	<code>\$a <= \$b</code>

Slika 4.2 Operatori poređenja

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Logički i operatori nad bitovima:

Logički operator	Ime	Primer	Rezultat
!	negacija	!\$a	Vraća <i>true</i> ako je \$a <i>false</i> i obrnuto.
&&	konjunkcija	\$a && \$b	Vraća <i>true</i> samo ako su \$a i \$b <i>true</i> .
	disjunkcija	\$a \$b	Vraća <i>true</i> ako je bar jedan operand <i>true</i> .
and	konjunkcija	\$a and \$b	Isto kao &&, ali sa nižim prioritetom.
or	disjunkcija	\$a or \$b	Isto kao , ali sa nižim prioritetom.

Opr. nad bitovima	Ime	Primer	Rezultat
&	konjunkcija	\$a & \$b	Bitovi aktivni u \$a i \$b aktivni su u rezultatu.
	disjunkcija	\$a \$b	Bitovi aktivni u \$a ili \$b aktivni su u rezultatu.
~	negacija	~ \$a	Bitovi aktivni u \$a nisu aktivni u rezultatu.
^	isključiva dijunkcija	\$a ^ \$b	Bitovi aktivni ili u \$a ili u \$b ali ne u oba, aktivni su u rezultatu.
<<	pomeranje ulevo	\$a << \$b	Pomera bitove \$a u levo za \$b mesta.
>>	pomeranje udesno	\$a >> \$b	Pomera bitove \$a u desno za \$b mesta.

Slika 4.3 Logički i operatori nad bitovima

Operatori za rad sa nizovima:

Operator	Ime	Primer	Rezultat
+	unija	\$a + \$b	Vraća niz koji se sastoji od svih elemenata \$a i \$b.
==	jednako	\$a == \$b	Vraća <i>true</i> ako \$a i \$b imaju jednake elemente.
===	identično	\$a === \$b	Vraća <i>true</i> ako \$a i \$b imaju jednake elemente u jednakom redosledu.
!=	različito	\$a != \$b	Vraća <i>true</i> ako je \$a različit od \$b.
<>	različito	\$a <> \$b	Vraća <i>true</i> ako je \$a različit od \$b.
!==	nije identično	\$a !== \$b	Vraća <i>true</i> ako \$a nije identičan \$b.

Slika 4.4 Operatori za rad sa nizovima

▼ Poglavlje 5

Upravljačke strukture u PHP-u

STRUKTURE GRANANJA

Strukture grananja omogućavaju da se u zavisnosti od trenutne vrednosti ili stanja određenih promenljivih i konstanti izvrše različiti delovi programa.

Strukture grananja

Strukture grananja omogućavaju da se u zavisnosti od trenutne vrednosti ili stanja određenih promenljivih i konstanti izvrše različiti delovi programa. **PHP** programski jezik podržava dve osnovne strukture grananja:

if/else i switch/case.

Upravljačke strukture omogućavaju upravljanje tokom izvršenja programa ili skripta.

Upravljačke strukture se mogu grupisati u:

- Uslovne strukture sa grananjem – konstrukcije koje programu omogućavaju donošenje odluka
- Iskaz if
- Iskaz else
- Iskazi elseif
- Iskaz switch
- Strukture sa ponavljanjem ili petlje – konstrukcije za ponavljanje određenog bloka koda u programu
- Petlja while
- Petlja for
- Petlja do... while

Uslovne strukture sa grananjem

Iskaz *if*. Upotrebljava se za donošenje odluke u zavisnosti od određenog uslova. Ako je uslov ispunjen (vraća *true*), izvršava se blok koda koji sledi iza iskaza *if* ; u suprotnom će blok biti preskočen. Uslovi se navode između zagrada.

Iskaz *else*. Definiše alternativnu akciju koju treba preduzeti ako uslov u if iskazu nije ispunjen. Iskaz *elseif* ili *else if*. Određuje redosled obrade više opcija. Zadaje se više uslova u datom redosledu, a program izvršava blok koda koji odgovara samo prvom ispunjenom uslovu.

Primeri **PHP** koda:

[illegible]

USLOVNE STRUKTURE SA GRANANJEM

PHP izvršava iskaze od odgovarajuće oznake case do iskaza break.

Iskaz **switch** . Omogućava da uslov ima više od dve vrednosti (u iskazu *if*, uslov može imati samo dve vrednosti - **true** ili **false**).

U iskazu *switch*, uslov može imati više različitih vrednosti, koje moraju biti skalarnog tipa (*integer*, *string* ili *float*).

Za svaku vrednost koja se želi obraditi, potrebno je napisati iskaz case, i eventualno podrazumevani iskaz case za sve ostale vrednosti.

PHP izvršava iskaze od odgovarajuće oznake `case` do iskaza *break*. Bez iskaza *break*, izvršavaju se svi iskazi iza iskaza `case`. Kada se dođe do *break*, izvršavanje se nastavlja iza iskaza *switch*.

Primer: ispitivanje koja reklama je privukla kupca

```
HTML kod
<select name="find">
<option value = "a">Ja sam obican potrosac.</option>
<option value = "b">Cuo sam na televiziji.</option>
</select>
switch($find) // PHP kod podrazumeva se da je učitana iz niza POST
{
case 'a' :
echo '<p>Obican potrosac.</p>';
break;
case 'b' :
echo '<p>TV reklame.</p>';
break;
default :
echo '<p>Ne zna se.</p>';
```

```
break;  
}
```

Strukture sa ponavljanjem

Petlja *while*. Izvršava blok koda dok god je zadati uslov ispunjen. Ova petlja se obično koristi kada ne znamo unapred koliko će iteracija biti potrebno.

Osnovna struktura petlje je:

```
while (uslov) izraz; Primer: prikaz brojeva od 1 do 5 num = 1; while( num <= 5){echo  
num.""; num++;}
```

Petlja for. Koristi se kada se zna koliko je iteracija potrebno.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

STRUKTURE SA PONAVLJANJEM

Petlja while. Izvršava blok koda dok god je zadati uslov ispunjen.

Osnovna struktura petlje je:

```
for (izraz1; uslov; izraz2) izraz3;
```

- izraz1 se izvršava jednom na početku i zadaje početnu vrednost brojača
- uslov se ispituje pre svake iteracije; ako nije ispunjen, petlja prestaje; uslov obično ispituje da li je brojač došao do neke granice
- izraz2 se izvršava na kraju svake iteracije; obično menja vrednost brojača
- izraz3 se izvršava jednom u svakoj iteraciji; to je obično blok koda

Primer: prikaz brojeva od 1 do 5

```
for ( i = 1; i <= 5; $i++) {
```

```
echo $i."";
```

```
}
```

Petlja while. Izvršava blok koda dok god je zadati uslov ispunjen. Ova petlja se obično koristi kada ne znamo unapred koliko će iteracija biti potrebno.

Osnovna struktura petlje je:

```
while (uslov) izraz;
```

Primer: prikaz brojeva od 1 do 5

```
num = 1; while( num <= 5){echo num.""; num++;}
```

Petlja for. Koristi se kada se zna koliko je iteracija potrebno.

Osnovna struktura petlje je:

```
for (izraz1; uslov; izraz2) izraz3;
```

- izraz1 se izvršava jednom na početku i zadaje početnu vrednost brojača
- uslov se ispituje pre svake iteracije; ako nije ispunjen, petlja prestaje; uslov obično ispituje da li je brojač došao do neke granice

STRUKTURE SA PONAVLJANJEM - NASTAVAK

Petlja do...while. Izvršava blok koda bar jednom, jer se uslov za izlazak iz petlje nalazi na njenom kraju.

- izraz2 se izvršava na kraju svake iteracije; obično menja vrednost brojača
- izraz3 se izvršava jednom u svakoj iteraciji; to je obično blok koda.

Primer: prikaz brojeva od 1 do 5

```
for ( i = 1; i <= 5; $i++) {  
    echo $i."  
";
```

Petlja **do...while**. Izvršava blok koda bar jednom, jer se uslov za izlazak iz petlje nalazi na njenom kraju.

Osnovna struktura petlje je:

```
do  
    izraz;  
while (uslov);
```

Primer: prikaz brojeva od 1 do 5

```
num = 1; do {echo num."  
";  
$num++;  
}
```

```
while ($num <= 5);
```

Izlazak iz upravljačke strukture. Postoje tri načina da se prekine izvršavanje bloka koda:

- iskaz break – izvršavanje skripta će se nastaviti od prvog reda iza petlje
- iskaz continue – započinje novu iteraciju petlje
- iskaz exit – prekida izvršavanje celog PHP skripta; koristi se kada se utvrdi neka greška

▼ Poglavlje 6

Upotreba PHP-a - izrada formi

VEZA HTML I PHP STRANICE

Jedna od najčešćih primena serverskih skript jezika je obrada HTML formi.

Kao i kod drugih jezika, kod u PHP-u se momože direktno pisati na tekućoj stranici ili “uvesti” kao spoljašnji fajl.

„Funkcija *include*, kompletan sadržaj zadatog spoljašnjeg fajla postavlja u stranicu koja je taj kod pozvala.

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

```
<?php?>
```

```
nclude("header.php");
```

```
<p> text</p>
```

```
</body>
```

```
</html>
```

„Ovaj kod se “kopira” na mesto gde je funkcija include delovala <html>

„Ovaj deo koda se tada ponaša kao sastavni deo početne stranice.

„Jedna od najčešćih primena serverskih skript jezika je obrada HTML formi. Da bi se forma napravila potrebno je:

1. Napraviti HTML stranicu sa tagom *form*, atributom *action* i *method*
2. U formularu napraviti elemente kojii imaju definisane *name* attribute
3. Napraviti taster *Submit* za slanje podataka na *php* stranu, koja je definisana u atributu *action*
4. Napraviti pomenutu PHP stranicu u kojoj se poziva *a = _POST[' atribut_name ']*, i podatak je iz HTML strane došao u PHPPPH stranicu u promenljivu *\$a* .

VEZA HTML I PHP STRANICE - IZRADA FORME

Prenos sadržaja tekstualnog polja u PHP se vrši po atributu name.

„**Prenos** sadržaja iz **tekstualnog polja** se obavlja na sledeći ančin:

- Prenos sadržaja tekstualnog polja u PHP se vrši po atributu *name*
- „ Iz tog razloga treba voditi računa da ne bude prklapanja atributa po imenu
- „Sadržaj polja se prenosi u originalu, bezz obzira na broj reči i razmake
- „Preuzimanje u PHP kodu je pomoću

\$_POST[' atribut_name ']

- „Kod nekih verzija dovoljno je samoo *\$atribut_name*

„**PRIMER 1:** Da bi se napravila forma sa slike u HTML-u i poslalo obaveštenje mejlom o primeljenim podacima:

Slika 6.1 Izgled jednostavne forme

Potrebno je u HTML napraviti:

```
<html>
<head>
<title>Formular za PHP testiranje</title>
</head>
<body>
<h2>Prost formular</h2>
<form method="post" action="formprocesor.php">
<input type="text" size="26" name="ime">Ime: <br/>

<input type="text" size="26" name="prezime">Prezime: <br/>
<input type="text" size="26" name="email">E mail adresa:<br/>
<textarea rows="4" cols="26" name="address"></textarea> Adresa: <br/>

<input type="submit" value="Send" name="submit">
<input type="reset" value="Clear" name="reset">
</form>
</body>
</html>
```

I u PHP fajlu napisati kod:

```
<?php
// šalje podatke i zahvaljuje korisniku
print "<h2>Hvala Vam $ime, primili smo vaše informacijeacije </h2>";
print "Ime: $ime <br/> \n";
print "Prezime: $prezime <br/> \n";
print "Email adresa: $email <br/> \n";
print "Adresa: $adresa <br/>\n";

// šalje podatke formulara na specificiranu adresu, email adresu

$to = "peraperic@NNscience.org";
$subject = "Informacije prikupljene formularom";
$body = "Ime: $ime \n Prezime: $prezime \n Email adresa: $email \n
```

```
Adresa: $adresa \n";  
  
mail ($to,$subject,$body);  
?>
```

KONTROLNI TASTERI - CHECK TASTER

U radu sa formama i na klijentskoj strani kod izbora više ponuđenih opcija elemenata na veb strani veoma je korisno imati kontrolne tastere kao što su Check (izaberi) i Radio (jedan izbor).

U radu sa formama i na klijentskoj strani kod izbora više ponuđenih opcija elemenata na veb strani veoma je korisno imati kontrolne tastere kao što su Check (izaberi) i Radio (jedan od ponuđenih).

Check taster (dugme)

Koristi se za:

- > Prenos sadržaja checkbox-a u PHP po atributu *name*
- > Svaki *box* postavljen je za sebe (neku od mogućih informacija)

Ako u tagu *<input>* nije definisan atribut *value*, tada je rezultat čekiranog polja u PHP-u "on" (uključeno - dok se u suprotnom ne prikazuje).

Ako je atribut *value* definisan, prenosi se *value* vrednost.

Za slučaj kada ima više *checkbox*-ova moguće je za *name* kreirati niz. Na taj način sva polja se prozivaju preko imena niza, čiji je element određen redosledom pojavljivanja *box*-ova (tastera).

Primer 2: Potrebno je napraviti fajl index.html sa sledećim kodom:

```
<html>  
<head> </head>  
<body>  
<form action="1.php" method="post">  
Opcije:  
1<input type="checkbox" name="1" >  
2<input type="checkbox" name="2" >  
3<input type="checkbox" name="3"  
value="Treci" >  
4<input type="checkbox" name="4"  
value="Cetvrti" ><br>  
<input type="submit" value="Obrada">  
</form>  
</body>  
</html>
```

Slika 6.2 Izgled veb stranice sa ponuđenim opcijama - Check tasteri

RADIO TASTER

Čest je slučaj da klijent želi izabrati jednu od ponuđenih opcija koje se međusobno isključuju. Tada se koristi Radio taster.

Nakon toga potrebno je napraviti fajl Primer 2.php sa sledećim kodom da bi se nakon čekiranja i izbora neke od opcija 1,2, 3 ili 4 izvršila dorada pritiskom na taster dorada i dobio odgovor na klijentskoj strani kao na slici 3.

```
<html>
<body>
Predmeti: <br>
<?php
echo $_POST['1'];
echo ("<br>");
echo $_POST['2'];
echo $_POST['3'];
echo $_POST['4'];
?>
</body></html>
```

Slika 6.3 Odgovor na klijentskoj strani

Čest je slučaj da klijent želi izabrati jednu od ponuđenih opcija koje se međusobno isključuju. Tada se koristi Radio taster.

Koristi se za:

- > Prenos sadržaja checkbox-a u PHP se vrši po atributu *name*
- > Svaki *radio button* mora imati isti *name*
- > Ako u tagu *<input>* nije definisan atribut *value*, tada je rezultat čekiranog polja u PHP-u on (dok se u suprotnom ne prikazuje)
- > Ako je atribut *value* definisan, prenosi se *value* vrednost.

Evo i Primera 3:

Potrebno je napraviti fajl *index.html* sa sledećim kodom:

```
<html>
<head> </head>
<body>
<form action="1.php" method="post">
Opcije:
1<input type="radio" name="1" >
2<input type="radio" name="1"
value="Drugi" > <br>
<input type="submit" value="Obrada">
</form>
```

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

RADIO TASTER - NASTAVAK

Izgled veb stranice na klijentskoj strani sa ponuđenim opcijama koje se isključuju rešavaju se upotrebom Radio tastera.

Izgled veb stranice na klijentskoj strani sa ponuđenim opcijama koje se isključuju - Radio tasteri i njihovi odgovori dobijeni od servera su dati na slici 4 i 5.

Izgled php koda je sledeći i zapisan je u fajlu Primer 3.php:

```
<html>  
<body>  
Izbor: <br>  
<?php  
echo $_POST['1'];  
?>  
</body>  
</html>
```

Slika 6.4 Izgled veb strane pre izbora i nakon izbora Radio tastera 1

Slika 6.5 Izgled veb strane pre izbora i nakon izbora Radio tastera 1

▼ Poglavlje 7

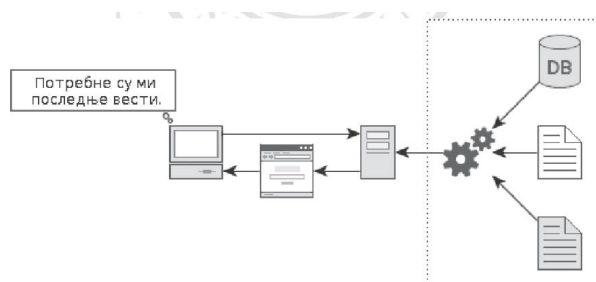
Izvršno PHP okruženje na strani servera

DINAMIČKO KREIRANJE SADRŽAJA

Začeci dinamičkog kreiranja sadržaja, odnosno programiranja na strani Veb servera, nalaze se kod korišćenja JavaScript kodova na strani servera (tzv. server side scripting).

U prvim danima Veba uloga servera bila je da sa spoljne memorije učitava traženi statični sadržaj (HTML dokument, sliku i drugo) i da ga u okviru HTTP protokola pošalje nazad klijentu. Takav princip rada nudi visoke performanse i bezbednost, ali su njegove mogućnosti veoma ograničene i svode se na gotovo jednosmernu komunikaciju, odnosno „emitovanje sadržaja“.

Dinamičko kreiranje sadržaja, sa druge strane, pristup je kod koga klijent serveru šalje standardan zahtev za određenim sadržajem, a taj zahtev se obrađuje odgovarajućom programskom procedurom na serveru. Rezultat te obrade je traženi sadržaj, kreiran u skladu sa trenutnom situacijom i specifičnostima definisanim u zahtevu. Ovakav pristup znatno umanjuje performanse i stvara mogućnost za bezbednosne probleme, ali nudi gotovo neograničene mogućnosti za razvoj mrežnih aplikacija.



Slika 7.1 Princip rada Veb servera sa dinamički kreiranim sadržajima

Začeci dinamičkog kreiranja sadržaja, odnosno programiranja na strani Veb servera, nalaze se kod korišćenja JavaScript kodova na strani servera (tzv. **server side scripting**). Kasnije je omogućeno korišćenje C, Perl i shell skriptova kroz CGI interfejs.

Najpopularnija rešenja za programiranje na strani servera danas su **-PHP_Hypertext Preprocessor, ASP - Active Server Pages i JSP - Java Server Pages**. ASP je komercijalna tehnologija kompanije Majkrosoft i ograničena je na upotrebu samo na operativnim sistemima te kompanije. JSP tehnologija je značajna jer omogućava upotrebu već postojećih klasa razvijenih u Java programskom jeziku, a slobodno je dostupna na različitim operativnim sistemima. Međutim, daleko najpopularnije rešenje za razvoj Veb aplikacija danas je PHP koji

je dostupan na različitim operativnim sistemima i uglavnom je deo standardne ponude Veb hosting provajdera.

Programski jezik PHP

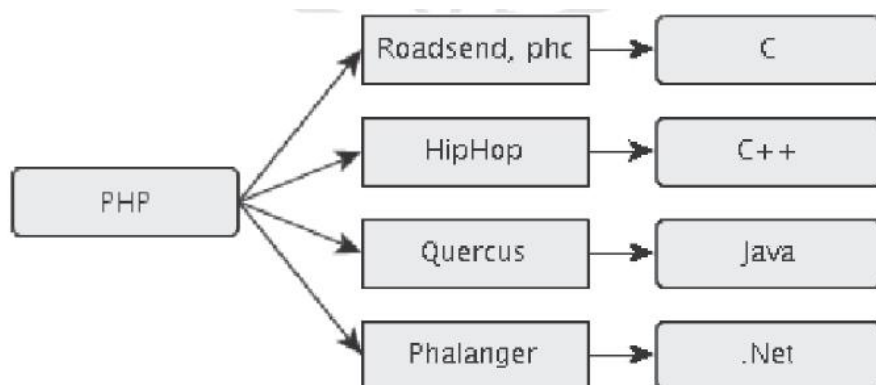
Sam **PHP** programski jezik se pojavio 1995. godine, a inicijalno je razvijen u **Perl** programskom jeziku i korišćen je kroz CGI interfejs. Njegov razvoj je pokrenuo Rasmus Lerdorf, tada 27-godišnji programer danskog porekla, a kasnije je nastavljen od strane Endija Gutmansa i Ziva Šuraskog, izraelskih programera, u okviru kompanije Zend Tehnologis.

IZVRŠNO OKRUŽENJE

Za potvrdu značaja i kvaliteta ovog programskog jezika dovoljno govori podatak da on pokreće preko 80% dinamičkih Veb sajtova na Internetu.

Tokom svog životnog veka je više puta prepisivan od nule a danas se razvija korišćenjem **C** programskog jezika. U četvrtoj verziji je dobio osnovnu podršku za objektno-orijentisano programiranje, a od pete verzije je ta podrška slična kao kod **Java** programskog jezika.

Do četvrte verzije **PHP** je bio interpretirani programski jezik. Međutim, 2000. godine je uveden Zend Endžin koji je izvršavanje PHP kodova podelio u nekoliko faza, od kojih je jedna prevođenje izvornog koda u opkod koji je sličan bajtkodu Java programskog jezika. Ovime je uvedena mogućnost keširanja opkoda čime je otvorena mogućnost za znatno podizanje performansi pri izvršavanju.



Slika 7.2 Postoji više rešenja za prevođenje PHP koda na druge jezike

S vremena na vreme se pojavljuju kritike **PHP** programskog jezika. U napadima na njega često se koriste argumenti koji su po ozbiljnosti i informisanosti kritičara bliski komentaru koji su dali voditelji izvesne brazilske TV stanice da „ukoliko se Veb adresa završava sa .php na njoj se obično nalazi virus“. Kritičari često napadaju dinamičnost tipova promenljivih kod PHP-a što, međutim, problem može predstavljati samo osrednjim programerima, dok je za početnike i eksperte ovakav tretman promenljivih nezamenljiv. Ukratko, za potvrdu značaja i kvaliteta ovog programskog jezika dovoljno govori podatak da on pokreće preko 80% dinamičkih Veb sajtova na Internetu kao i da se on koristi za neke od najvećih platformi kao što su Fejsbuk i Vikipedija.

Izvršno okruženje

Primarni način izvršavanja programa napisanih u **PHP** programskom jeziku je u okviru obrade zahteva ka Veb serveru. Ovakav način izvršavanja podrazumeva postojanje **PHP** interpretera/ izvršioca unutar Veb servera, kome se prosleđuju svi učitani fajlovi sa spoljne memorije koji imaju ekstenziju `.php`. Iz ovih fajlova se uzimaju delovi koji se nalaze između oznaka:

`<?php i ?>` i kod unutar njih se interpretira i izvršava.

KONFIGURACIONI FAJLOVI

Treba imati u vidu da je ispravno podešen PHP uslov za nesmetano i ispravno izvršavanje Veb aplikacija na serveru, kao i da propusti u konfiguraciji mogu otvoriti bezbednosne probleme.

Rezultati izvršavanja se uključuju na mesta na kojima se nalazio izvršeni blok izvornog koda ili se smeštaju u zaglavlje HTTPR paketa ukoliko je tako zahtevano.

Osim u okviru obrade zahteva ka Veb serveru PHP programi se mogu izvršavati i samostalno - direktnim povezivanjem putem alfa-numeričke ljuške operativnog sistema - ili u okviru grafičkih desktop aplikacija. Međutim, ovi pristupi su daleko ređe korišćeni.

Konfiguracioni fajlovi

Dva osnovna fajla kojim se uključuje i podešava korišćenje PHPR-a na Veb serveru jesu konfiguracioni fajl samog Veb servera (gde se uključuje korišćenje samog **PHP**-a) i konfiguracioni fajl **PHP**-a kojim se definiše njegovo ponašanje. U nastavku je dat sadržaj konfiguracionog fajla `mod_php.conf` kao dela konfiguracije Apač Veb servera kojim se uključuje podrška za **PHP**:

```
# mod_php - PHP Hypertext Preprocessor module
# Load the PHP module:
LoadModule php5_module lib/httpd/modules/libphp5.so
# Tell Apache to feed all *.php files through PHP.
<FilesMatch \.php$>
SetHandler application/x-httpd-php
</FilesMatch>
```

U okviru fajla `php.ini` definišu se parametri koji određuju kako će se **PHP** ponašati - koliko će memorije i vremena biti dostupno za izvršavanje, koja proširenja su uključena i slično. Ovi parametri se definišu u formatu:

parametar = vrednost

U nastavku je dat primer tipičnih konfiguracionih parametara PHP-a:

`Trmax_execution_time = 180 memory_limit = 128M display_errors = Off file_uploads = On extension=mysqli.so extension=zip.sodate.timezone = "Europe/Belgrade"`

Treba imati u vidu da je ispravno podešen PHP uslov za nesmetano i ispravno izvršavanje Veb aplikacija na serveru, kao i da propusti u konfiguraciji mogu otvoriti bezbednosne probleme.

Takođe, postoje značajne razlike u konfiguraciji između razvojnih i produkcionih okruženja, kao i između različitih produkcionih okruženja.

UPRAVLJANJE GREŠKAMA

Upozorenja (engl. warning) su ozbiljnije greške, za koje je teško pretpostaviti da spadaju u planirano ponašanje aplikacije, ali je i nakon njih moguće nastaviti izvršavanje skripta.

Tri osnovna tipa grešaka kod izvršavanja **PHP** skriptova su opomena, upozorenje i fatalna greška. Opomene (engl. *notice*) su manje greške koje mogu dovesti do problema u izvršavanju ali je moguće i da je takav pristup namerno korišćen od strane programera (na primer, pokušaj da se pristupi nepostojećem atributu promenljive koja nije objekat).

```
[Wed Jul 17 22:19:03.626987 2013] [:error] [pid 2494:tid 2961177408] [client 127.0.0.1:38383] PHP Notice: Trying to get property of non-object in /home/ajevremovic/publichtml/Lab/knjiga-it/db/izmena.php on line 38, referer: http://localhost/Lab/knjiga-it/db/select.php
```

Upozorenja (engl. *warning*) su ozbiljnije greške, za koje je teško pretpostaviti da spadaju u planirano ponašanje aplikacije, ali je i nakon njih moguće nastaviti izvršavanje skripta. U nastavku je dat primer upozorenja vezanog za pokušaj dodavanja zaglavlja u HTTP paket u trenutku kada je već započeto slanje njegovog sadržaja:

```
[Fri Dec 07 23:40:23.249784 2012] [:error] [pid 3213:tid 2844785472] [client 127.0.0.1:54438 ]
```

```
PHP Warning: Cannot modify header information - headers already sent by (output started at /ws/s1.php:468) in /ws.s1.php on line 467
```

Fatalne greške (engl. **fatal error**) su najozbiljniji tip grešaka posle čijeg javljanja se dalje izvršavanje prekida jer se smatra da nije moguće ispravno nastaviti i dobiti očekivane rezultate. U nastavku je dat primer poruke o fatalnoj grešci, nastaloj kao posledica poziva funkcije koja nije definisana:

```
[ Sun Dec 09 22:30:53.911059 2012] [:error] [pid 2470:tid 2995780416] [client 127.0.0.1:58113] PHP Fatal error: Call to undefined function lzfdcompress() in /home/ajevremovic/publichtml/kompresija.php on line 25
```

Informacije o greškama se mogu prikazati direktno ili preusmeriti u fajl sa dnevnikom događaja. U razvojnim okruženjima je direktno prikazivanje bolji izbor jer programeru odmah daje informaciju o greškama koje se javljaju, dok prikazivanje grešaka korisnicima u produkcionom okruženju može dovesti do bezbednosnih problema.

UPRAVLJANJE GREŠKAMA - NASTAVAK

Informacije o greškama se mogu prikazati direktno ili preusmeriti u fajl sa dnevnikom događaja.

Da li će se greške prikazivati ili skladištiti u fajl određuje se sledećim direktivama u konfiguracionom fajlu:

```
display_errors = On / Off log_errors = On / Off
```

Ukoliko je uključeno čuvanje informacija o greškama u fajlu sa dnevnikom događaja, podrazumevano će se koristiti fajl `eggoglod` samog Veb servera. Jedan način da se informacija o nastaloj grešci pri izvršavanju funkcije ili iskaza ne prikaže niti zabeleži u fajlu jeste korišćenje operatora „@“:

```
x = 1; echo@ x->lme;
```

Međutim, ovakav način odbacivanja poruka o greškama treba veoma oprezno koristiti jer on može dovesti do drastično otežanog pronalaženja i otklanjanja grešaka u složenim aplikacijama.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 8

PHP i baze podataka

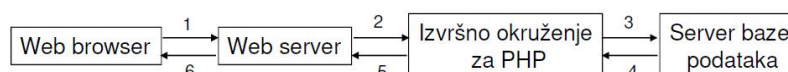
PRETRAŽIVANJE BAZE PODATAKA

Osnovna arhitektura sistema za pristup bazama podataka iz Web aplikacija sastoji se od Web browser-a, Web servera, mašine za izvršavanje skriptova i servera baze podataka.

Osnovna arhitektura sistema za pristup bazama podataka iz Web aplikacija sastoji se od Web browser-a, Web servera, mašine za izvršavanje skriptova i servera baze podataka (Slika 1).

Tipična transakcija baze podataka na Web-u sastoji se od sledećih koraka:

1. Korisnikov Web browser šalje HTTP zahtev za određenu Web stranicu (.php)
2. Web server prima zahtev za prikazivanje stranice, učitava datoteku u kojoj se ona nalazi i sadržaj datoteke prosleđuje izvršnom okruženju na obradu.
3. PHP-ovo izvršno okruženje počinje sintaksnu analizu skripta. Ako skript sadrži naredbe za uspostavljanje veze sa bazom podataka, PHP otvara vezu sa serverom baze i šalje mu odgovarajući upit.
4. Server baze prima upit, obrađuje ga i rezultat vraća izvršnom okruženju.



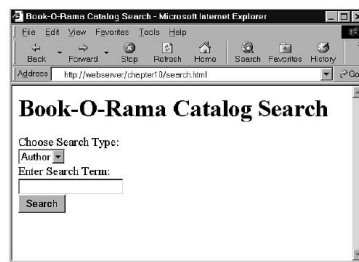
Slika 8.1 Osnovna arhitektura sistema za pristup bazama podataka iz Web aplikacija

5. PHP-ovo izvršno okruženje završava obradu skripta, što podrazumeva i formatiranje rezultata upita u HTML kod koji šalje Web serveru.
6. Dobijeni HTML kod, Web server prosleđuje čitaču koji ga prikazuje.

Pretraživanje baze podataka

Primer: P pretraživanje baze podataka koja sadrži katalog knjiga preuzeto iz reference: Luke Welling, Laura Thomson, PHP i MySQL: Razvoj aplikacija za Web, prevod Mikro knjiga, 2006.

Najpre treba omogućiti korisniku da unese podatke po kojima će se pretraživati, tj. treba generisati HTML stranicu sa formom za unos podataka. Kada korisnik unese kriterijum pretrage (u ovom primeru to može biti autor, ime knjige ili isbn kod knjige) i ključnu reč, pritiskom na dugme **Search** poziva se skript results.php koji se nalazi na Web serveru. Izgled stranice za pretraživanje:



Slika 8.2 Izgled stranice za pretraživanje

STRANICA ZA PRETRAŽIVANJE BAZE PODATAKA - SEARCH.HTML

Pritiskom na dugme Search poziva se skript results.php koji se nalazi na Web serveru. Izgled stranice za pretraživanje.

Stranica za pretraživanje baze podataka: search.html kod je:

```
<html>
<head>
<title>Book-O-Rama Catalog Search</title>
</head>
<body>
<h1>Book-O-Rama Catalog Search</h1>
<form action="results.php" method="post">
Choose Search Type:<br/>
<select name="searchtype">
<option value="author">Author</option>
<option value="title">Title</option>
<option value="isbn">ISBN</option>
</select>
<br/>
Enter Search Term:<br/>
<input name="searchterm" type="text">
<br/>
<input type="submit" value="Search">
</form>
</body>
</html>
```

PHP skript: results.php je:

```
<html>
<head>
<title>Book-O-Rama Search Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
// napravi promenljive s kratkim imenima
```

```

$searchtype=$_POST['serachtype'];
$searchterm=$_POST['serachterm'];
$searchterm=trim($serachterm);
if (!$searchtype || !$searchterm) {
echo 'You have not entered search details.'.
' Please go back and try again.';
exit;
}
if (!get_magic_quotes_gpc()) {
$searchtype=addslashes($searchtype);
$searchterm=addslashes($searchterm);
}
@ $db = new mysqli('localhost', 'bookorama',
'bookorama123', 'books');
if (mysqli_connect_errno()) {
echo 'Error: Could not connect to database.'.
' Please try again later.';
exit; }
$query = "select * from books where
".$searchtype." like'%" . $searchterm . "%'";
$result = $db->query($query);
$num_results = $result->num_rows;
echo '<p>Number of books found: ' $num_results.'</p>';
for ($i=0; $i < $num_results; $i++) {
$row = $result->fetch_assoc();
echo '<p><strong>.( $i+1 ).'. Title: ' ;
echo 'htmlspecialchars(stripslashes($row['title']))';
echo '</strong><br>Author: ' ;
echo 'stripslashes($row['author']);
echo '<br> ISBN: ' ;
echo 'stripslashes($row['isbn']);
echo '<br> Price: ' ;
echo 'stripslashes($row['price']);
echo '</p>';
}
$result->free();
$db->close ();
?>
</body>
</html>

```

SADRŽAJ SKRIPTA

Pre nego što se uspostavi veza sa bazom podataka, potrebno je proveriti ispravnost podataka koje je korisnik uneo.

U svakom skriptu koji pristupa bazi podataka sa Web-a treba da postoji odgovarajući kod za sledeće osnovne korake:

1. Provera podataka koje je korisnik uneo.

2. Filtriranje ulaznih podataka.
3. Uspostavljanje veze sa odgovarajućom bazom podataka.
4. Zadavanje upita toj bazi podataka.
5. Učitavanje rezultata.
6. Prikaz rezultata korisniku.

Provera ulaznih podataka

Pre nego što se uspostavi veza sa bazom podataka, potrebno je proveriti ispravnost podataka koje je korisnik uneo.

Provera podataka obuhvata sledeće korake:

a) Najpre treba ukloniti beline koje je korisnik možda uneo. To se radi funkcijom `trim()` koja se primenjuje na ulazne promenljive.

```
$searchterm=trim($searchterm);
```

b) Zatim treba proveriti da li je korisnik uneo pojam koji traži i kriterijum pretrage.

Ovde se vidi važnost prethodnog koraka, jer pojam koji se sastoji samo od beline funkcija `trim()` briše.

```
if (!$searchtype || !$searchterm) {  
    echo 'You have not entered search details.'  
    ' Please go back and try again.'  
    exit;  
}
```

Filtriranje ulaznih podataka

Da bi se ispravno obradili podaci koje korisnik šalje, neophodno je filtrirati upravljačke znakove (`"`, `'`, `\` i znak `NULL`) iz tih podataka. PHP podržava više funkcija za formatiranje stringova radi unošenja u bazu podataka.

Pre nego što se proslede bazi podataka, svi podaci koje korisnik šalje moraju se obraditi pomoću sledećih funkcija:

a) `addslashes()`. Ova funkcija dodaje znak `\` ispred svakog upravljačkog znaka. Na primer, nakon primene ove funkcije `"` postaje `\`, a `\` postaje `\\`. Argument funkcije je string, a rezultat je formatiran string.

FILTRIRANJE ULAZNIH PODATAKA

Pre prikazivanja korisniku, podatke dobijene iz baze podataka takođe treba filtrirati, kako bi se iz njih izbacio znak \ unet prilikom filtriranja ulaznih podataka.

b) `get_magic_quotes_gpc()`. PHP može biti podešen tako da automatski dodaje i uklanja \.

Provera da li je sistem tako podešen obavlja se pomoću navedene funkcije (tzv. mehanizam “magičnih navodnika”). Sufiks `gpc` je skraćenica za GET, POST i COOKIES,

što znači da se automatski konvertuju promenljive iz tih izvora.

```
if (!get_magic_quotes_gpc()) {  
    $searchtype=addslashes($searchtype);  
    $searchterm=addslashes($searchterm);  
}
```

Filtriranje rezultata

Pre prikazivanja korisniku, podatke dobijene iz baze podataka takođe treba filtrirati, kako bi se iz njih izbacio znak \ unet prilikom filtriranja ulaznih podataka, i izbegao pogrešan rad baze zbog specijanih znakova u HTML-u.

Pre nego što se prikažu korisniku, podatke treba obraditi pomoću sledećih funkcija:

a) `stripslashes()`. Ova funkcija izbacuje znak \ iz zadatog stringa.

Njome se podaci moraju obraditi ukoliko je mehanizam “magičnih navodnika” bio uključen.

b) `htmlspecialchars()`. Ova funkcija se koristi za pretvaranje znakova koji u HTML-u imaju specijalno značenje (&, “, < i >) u HTML entitete. Na primer, znak < pretvara se u <.

Uspostavljanje veze sa bazom

PHP5 ima novu biblioteku funkcija za uspostavljanje veza sa bazama podataka, koja se zove `mysqli` (i od *improved*). Ova biblioteka podržava upotrebu i objektno-orijentisane i proceduralne sintakse.

Objektno-orijentisani oblik sintakse je:

```
@ $db = new mysqli('localhost', 'bookorama', 'bookorama123', 'books');
```

Na ovaj način je napravljena instanca klase `mysqli` i uspostavljena veza sa serverom ‘localhost’ pod korisničkim nalogom ‘bookorama’ i lozinkom ‘bookorama123’, i određena je baza podataka ‘books’ sa kojom će se raditi. Ovaj oblik omogućava da se sa bazom radi tako što se pozivaju metodi objekta `$db`.

USPOSTAVLJANJE VEZE SA BAZOM

Broj istovremenih veza koje se mogu uspostaviti sa MySQL serverom je ograničen parametrom `max_connections` koji se podešava u datoteci `my.conf`.

Proceduralni oblik sintakse je:

```
@ $db = mysqli_connect('localhost', 'bookorama', 'bookorama123', 'books');
```

Ova funkcija vraća rezultat tipa resurs, koji predstavlja vezu sa bazom podataka. Ovaj resurs mora se proslediti svim ostalim funkcijama iz navedene biblioteke koje se pozivaju u kodu.

Napomena: Simbol @ predstavlja operator zanemarivanja greške i može se upotrebiti ispred svakog izraza. Ako se ovaj operator upotrebi, upozorenje o greškama se zanemaruje. Tada bi obavezno trebalo napisati kod za obradu grešaka.

Rezultat pokušaja uspostavljanje veze sa bazom podataka se uvek mora proveriti u kodu, jer ako on nije odgovarajući, preostali deo skripta neće raditi. Blok koda za proveru uspešnosti konekcije sa bazom je:

```
if (mysqli_connect_errno()) {  
    echo 'Error: Could not connect to database.'  
    ' Please try again later.';  
    exit;  
}
```

Funkcija `mysqli_connect_errno` vraća broj greške u slučaju neuspešnog povezivanja, odnosno 0 u slučaju uspešnog.

Broj istovremenih veza koje se mogu uspostaviti sa MySQL serverom je ograničen parametrom `max_connections` koji se podešava u datoteci `my.conf`.

Zadavanje upita bazi podataka

Pre izvršavanja upita, preporučuje se njegovo definisanje u vidu stringa:

```
$query = "select * from books where $searchtype like '%$searchterm%'";
```

U ovom upitu tražimo pojam koji je korisnik zadao (`$searchterm`) u polju koje je izabrao (`$searchtype`).

U objektno-orijentisanoj verziji, izvršavanje upita se pokreće pomoću funkcije:

```
$result = $db->query($query);
```

Funkcija vraća objekat koji omogućava čitanje rezultata upita.

ZADAVANJE UPITA BAZI PODATAKA

Postoji više funkcija koje omogućavaju da se pomoću objekta ili identifikatora rezultata učitaju rezultati upita.

U proceduralnoj verziji, upit se izvršava komandom:

```
$result = mysqli_query($db, $query);
```

Funkcija vraća identifikator tipa resurs.

U oba slučaja, ako se funkcija nije uspešno izvršila, rezultat je logička vrednost *false*.

Učitavanje rezultata upita

Postoji više funkcija koje omogućavaju da se pomoću objekta ili identifikatora rezultata učitaju rezultati upita. Objekat, odnosno identifikator rezultata je osnova za pristupanje redovima koje je upit vratio.

Kada se koristi objektno-orijentisani oblik, broj redova koje je upit vratio smešta se u svojstvo `num_rows` objekta rezultata (`$result`), kome se pristupa na sledeći način:

```
$num_results = $result->num_rows;
```

Kada se koristi proceduralni oblik, funkcija `mysqli_num_rows()`, čiji je parametar identifikator rezultata, vraća broj redova:

```
$num_results = mysqli_num_rows($result);
```

Broj vraćenih redova je koristan podatak ako treba obraditi ili prikazati rezultate.

Ako je on poznat, može se upotrebiti petlja:

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

```
{ // obrada rezultata }
```

Da bi se pročitao jedan red iz skupa rezultata, u okviru *for* petlje se pozivaju funkcije `$result->fetch_assoc()` (kod objektno-orijentisanog pristupa) ili `mysqli_fetch_assoc()` (kod proceduralnog pristupa):

```
$row = $result->fetch_assoc(); ili $row = mysqli_fetch_assoc($result);
```

Navedene funkcije čitaju svaki red iz skupa rezultata i vraćaju podatke u obliku asocijativnog niza, u kome je svaki ključ (indeks) ime jednog atributa, a svaka vrednost u nizu je odgovarajuća vrednost učitana iz baze.

Niz `$row` omogućava da se svako njegovo polje prikaže u odgovarajućem obliku:

```
echo '</strong><br>Author: ';  
echo 'stripslashes($row['author']);  
echo '<br> ISBN: ';  
echo 'stripslashes($row['isbn']);
```

```
echo '<br/> Price: ';  
echo 'stripslashes($row['price']);
```

UČITAVANJE REZULTATA UPITA

Red podataka iz skupa rezultata se može vratiti i u vidu niza sa numeričkim indeksima.

Red podataka iz skupa rezultata se može vratiti i u vidu niza sa numeričkim indeksima. U tu svrhu se koriste sledeće funkcije:

```
$row = $result->fetch_row();
```

ili

```
$row = mysqli_fetch_row($result);
```

U ovom slučaju, vrednosti atributa će se nalaziti u elementima niza `$row[0]`, `$row[1]`, itd.

Red podataka se može učitati i u obliku objekta na sledeći način:

```
$row = $result->fetch_object();
```

ili

```
$row = mysqli_fetch_object($result);
```

Svatom atributu se onda pristupa primenom sintakse: `$row->title`, `$row->author`, itd.

Prekidanje veze sa bazom

Skup rezultata se oslobađa pozivanjem metoda:

```
$result->free();
```

ili `mysqli_free_result($result);`

Zatim se za zatvaranje baze može koristiti metod:

```
$db->close();
```

ili funkcija

```
mysqli_close($db);
```

Pozivanje funkcija za zatvaranje baze nije obavezno, jer će se veza ionako prekinuti po izvršenju skripta.

Unos podataka u bazu

Unošenje podataka u bazu je vrlo slično čitanju podataka iz nje. Osnovni koraci su isti, tj. uspostavlja se veza sa bazom, šalje upit i ispituju rezultati. Razlika je u tome što se sada umesto upita tipa `SELECT`, šalje upit tipa `INSERT`.

Stranica za unošenje novih knjiga u bazu: newbook.html

```
<html>
<head>
<title>Book-0-Rama New Book Entry</title>
</head>
<body>
<h1>Book-0-Rama New Book Entry</h1>
<form action="insert_book.php" method="post">
<table border="0">
<tr>
<td>ISBN</td>
<td><input type="text" name="isbn"></td>
</tr>
<tr>
<td>Author</td>
<td><input type="text" name="author"></td>
</tr>
<tr>
<td>Title</td>
<td><input type="text" name="title"></td>
</tr>
<tr>
<td>Price</td>
<td><input type="text" name="price"></td>
</tr>
<tr>
<td colspan="2"><input type="submit"
value="Register"></td>
</tr>
</form>
</body>
</html>
```

UNOS PODATAKA U BAZU

Podaci iz ove forme prosleđuju se skriptu insert_book.php koji čita dobijene podatke, obavlja provere i pokušava da upiše podatke u bazu.

Stranica za unošenje novih knjiga u bazu: newbook.html daje formu za unos kao na slici 3.

Podaci iz ove forme prosleđuju se skriptu insert_book.php koji čita dobijene podatke, obavlja provere i pokušava da upiše podatke u bazu.

PHP skript: insert_book.php je

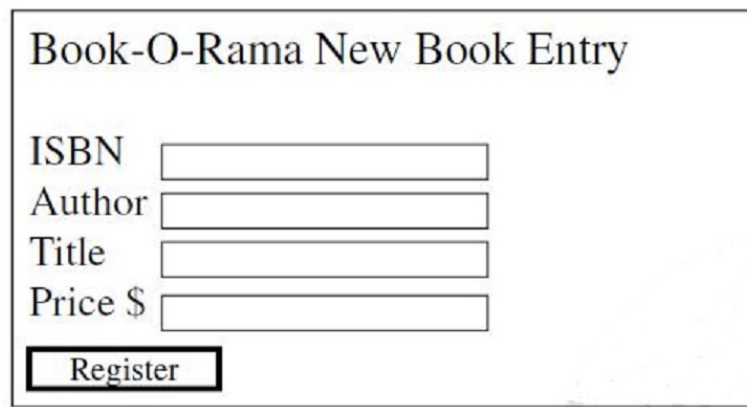
```
<html>
<head>
<title>Book-0-Rama Book Entry Results</title>
</head>
<body>
```

```
<h1>Book-0-Rama Book Entry Results</h1>
<?php
$isbn=$_POST['isbn'];
$author=$_POST['author'];
$title=$_POST['title'];
$price=$_POST['price'];
if (!$isbn || !$author || !$title || !$price) {
echo 'You have not entered all the required details.'.
' Please go back and try again.';
exit;
}
if (!get_magic_quotes_gpc()) {
$isbn=addslashes($isbn);
$author=addslashes($author);
$title=addslashes($title);
$price=doubleval($price);
}
@ $db = new mysqli('localhost', 'bookorama',
'bookorama123', 'books');
if (mysqli_connect_errno()) {
echo 'Error: Could not connect to database.'.
' Please try again later.';
exit;
}
$query = "insert into books values
(' ".$isbn." ',' ".$author." ',' ".$title." ',' ".$price." ')";
$result = $db->query($query);
if ($result)
echo $db->affected_rows.' book inserted into database.';
$db->close ();
?>
</body>
</html>
```

Kod skripta `insert_book.php` je sličan kodu skripta za pretraživanje baze podataka. On implementira sledeće korake:

1. Najpre se proverava da li je korisnik popunio sva polja u formi.

```
if (! $isbn || ! $author || ! $title || ! $price) {
echo 'You have not entered all the required details.'.
' Please go back and try again.';
exit;}
```



Slika 8.3 Izgled stranice za unos

UNOS PODATAKA U BAZU - NASTAVAK

Filtriranje nepotrebnih znakova u numeričkom polju postiže se primenom funkcije `dbleval()`.

2. Zatim se uneti podaci formatiraju u oblik pogodan za unos u bazu podataka.

```
if (!get_magic_quotes_gpc()) {  
    $isbn=addslashes($isbn);  
    $author=addslashes($author);  
    $title=addslashes($title);  
    $price=dbleval($price);} 
```

S obzirom da se u bazi cena čuva u formatu pokretnog zareza, kose crte nisu potrebne tom podatku.

Filtriranje nepotrebnih znakova u numeričkom polju postiže se primenom funkcije `dbleval()`.

3. Upostavlja se konekcija sa bazom podataka.

4. Definiše se upit koji se šalje bazi podataka.

```
$query = "insert into books values (' ".$isbn." ',' ".$author." ',' ".$title." ',' ".$price." ');"  
$result = $db->query($query);
```

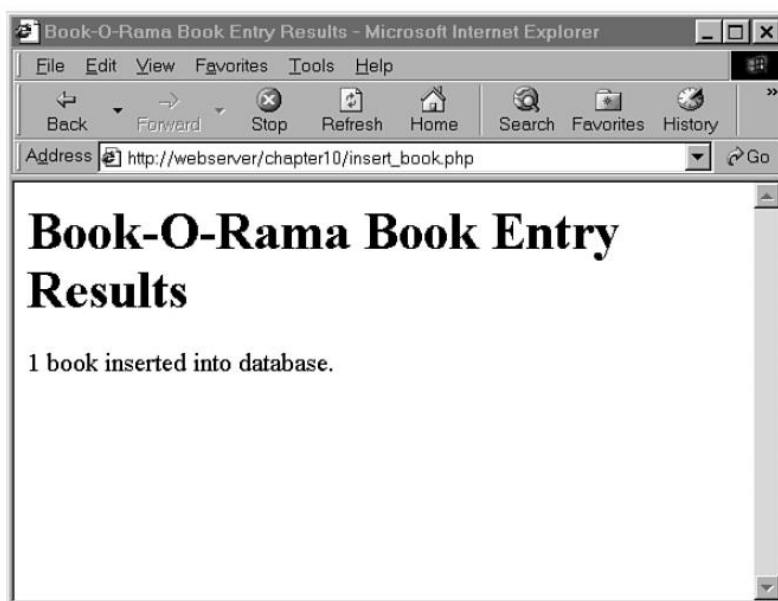
Značajna razlika između upotrebe upita INSERT u odnosu na SELECT je u načinu korišćenja funkcije `mysqli_affected_rows()` (u proceduralnoj verziji) ili u objektno-orijentisanoj verziji

```
echo $db->affected_rows.' book inserted into database.';
```

U prethodnom skriptu smo koristili funkciju `mysqli_num_rows()` da bismo saznali koliko redova je vratio upit SELECT. Ako upit menja stanje u bazi (INSERT, DELETE i UPDATE) treba koristiti

```
funkciju mysqli_affected_rows() .
```

Izgled stranice nakon izvršenja skripta insert_book.php je:



Slika 8.4 Izgled stranice nakon izvršenja skripta insert_book.php

▼ Poglavlje 9

Pokazna Vežba 7

OSNOVE PHP PROGRAMSKOG JEZIKA - UVOD

Cilj ove sekcije je da studenta uvede u vežbu

PHP je skripting jezik koji se izvršava na serverskoj strani i predstavlja jako moćan alat za dinamičke i interaktivne web strane i aplikacije. PHP ima široku upotrebu, potpuno je besplatan i efikasan u odnosu na druge serverske skripting jezike poput ASP jezika.

Primer jedne **PHP** skripte

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "Moja prva PHP skripta!";
?>

</body>
</html>
```

Svaki php kod počinje sa znakom za manje, upitnikom i tekstom php naslovom **<?php** što označava početak koda a završava se upitnikom i znakom veće. **?>**

PHP jezik je veoma sintakasno sličan C-u. PHP 5 ne podržava tipiziranje promenjivih dok je to moguće u PHP-u 7.

Primer promenjivih u PHP-u 5.

```
//brojna promenjiva
$i = 20;
$b = 20.2;
//tekstualna promenjiva
$text = "Primer teksta";
$text = 'Primer teksta';
```

Petlje u PHP-u

PHP podržava sledeće petlje.

- Foreach
- For
- While
- Do while

Primer for petlje:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "Broj je: $x <br>";
}
?>
```

Primer **foreach** petlje:

```
<?php
$broj = array("red", "green", "blue", "yellow");
foreach ($broj as $vrednost) {
    echo "$vrednost <br>";
}
?>
```

Primer **while** petlje:

```
<?php
$x = 1;
while($x <= 5) {
    echo "Broj je: $x <br>";
    $x++;
}
?>
```

Primer **do while** petlje:

```
<?php
$x = 1;
do {
    echo "Broj je: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

FUNKCIJE U PHP-U

Cilj ove sekcije je da pokaže korišćenje funkcija u PHP-u

Funkcije u PHP-u se koriste jako jednostavno. U funkcijama nije potrebno definisati return type već ukoliko se pojavi return unutar funkcije to će biti funkcija koja vraća parametar dok ako ne postoji return ključna reč u metodi ona neće vratiti ništa (void).

```
<?php
function ispisiPoruku() {
    echo "Hello world!";
}
ispisiPoruku(); // poziv funkcije
?>
```


Funkcija takođe može imati argumente. Pošto PHP nije tipiziran sve što treba da u funkciji je da navedemo imena promenljivih koje ćemo primiti kao argument funkcije. Primer

```
<?php
function imeZaPrezime($fname) {
    echo "$fname Vasic.<br>";
}
imeZaPrezime("Vuk");
imeZaPrezime("Dejan");
imeZaPrezime("Selen");
imeZaPrezime("Saša");
imeZaPrezime("Igor");
?>
```

Na sledećem primeru je funkcija koja ima dva parametra.

```
<?php
function imeZaPrezime($fname, $year) {
    echo "$fname Vasic. Rođen je u godini $year <br>";
}
imeZaPrezime("Žika", "1975");
imeZaPrezime("Pera", "1978");
imeZaPrezime("Laza", "1983");
?>
```

Argumenti u funkcijama mogu imati i predefinisane vrednosti

```
<?php
function setHeight($minheight = 50) {
    echo "Visina je : $minheight <br>";
}

setHeight(350);
setHeight(); // ispisace 50
setHeight(135);
setHeight(80);
?>
```

Ispod se nalazi primer funkcije koje vraća zbir dva broja. (koristi return)

```
<?php
function suma($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . suma(5, 10) . "<br>";
echo "7 + 13 = " . suma(7, 13) . "<br>";
echo "2 + 4 = " . suma(2, 4);
?>
```

NIZOVI U PHP-U

Cilj ove sekcije je da prikaže korišćenje nizova u PHP-u

Nizovi omogućavaju da smestite više podataka u jednu promenjivu. Primer korišćenja nizova u PHP-u je sledeći:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Funkcijom count možemo izbrojati koliko ima elemenata tačno u nizu.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Korišćenje petlje u nizovima u PHP-u.

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arlength = count($cars);

for($x = 0; $x < $arlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

SERVISI U PHP-U

Cilj ove sekcije je da pokaže studentu kako može napraviti servis u PHP-u

U PHP-u je moguće praviti GET i POST web servise koristeći ili XML ili JSON. XML možemo. Na primer ukoliko želimo da napravimo GET servis koji vraća XML to možemo uraditi na sledeći način:

```
<?php
header("Content-type: text/xml");
$test_array = array (
    'Vuk' => 'ime',
    'Vasic' => 'prezime',
    '177894488885798' => 'jmbg',
);
$xml = new SimpleXMLElement('<root/>');
```

```
array_walk_recursive($test_array, array ($xml, 'addChild'));  
print $xml->asXML();  
?>
```

Servis na GET vraća:

```
<root>  
<ime>Vuk</ime>  
Vasic</prezime>  
<jmbg>177894488885798</jmbg>  
</root>
```

Ukoliko bi hteli isto da ponovimo samo da je u pitanju JSON servis možemo to uraditi na sledeći način:

```
<?php  
header("Content-type: application/json");  
$test_array = array (  
    'ime' => 'Vuk',  
    'prezime' => 'Vasic',  
    'jmbg' => '177894488885798',  
);  
echo json_encode($test_array);  
?>
```

Servis na GET vraća:

```
{"ime":"Vuk","prezime":"Vasic","jmbg":"177894488885798"}
```

Takođe možemo napraviti i jedan servis koji omogućava preko GET parametra da vrati ili JSON ili XML u zavisnosti od potrebe. Primer

```
<?php  
$type = $_GET['type'];  
if($type == "json"){  
    header("Content-type: application/json");  
    $test_array = array (  
        'ime' => 'Vuk',  
        'prezime' => 'Vasic',  
        'jmbg' => '177894488885798',  
    );  
    echo json_encode($test_array);  
}else {  
    header("Content-type: text/xml");  
    $test_array = array (  
        'Vuk' => 'ime',  
        'Vasic' => 'prezime',  
        '177894488885798' => 'jmbg',  
    );  
    $xml = new SimpleXMLElement('<root/>');  
    array_walk_recursive($test_array, array ($xml, 'addChild'));  
    print $xml->asXML();  
}
```

```
}  
  
?>
```

Ukoliko prosledimo na ovaj servis type parameter i vrednost json dobićemo:

```
{"ime":"Vuk","prezime":"Vasic","jmbg":"177894488885798"}
```

Dok ako prosledimo xml dobijamo:

```
<root>  
<ime>Vuk</ime>  
Vasic</prezime>  
<jmbg>177894488885798</jmbg>  
</root>
```

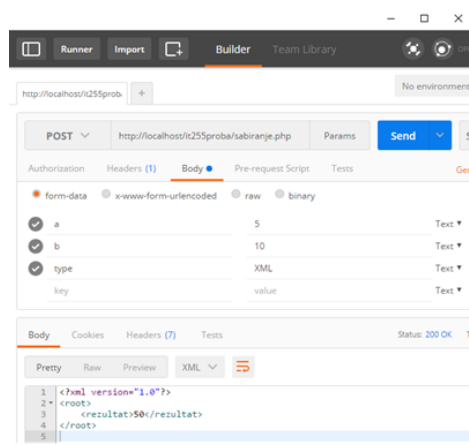
PRIMER WEB SERVIS KOJI RADI OPERACIJU SA DVA BROJA

Cilj ove sekcije je da prikaže studentu kako napraviti servis koji radi neku operaciju

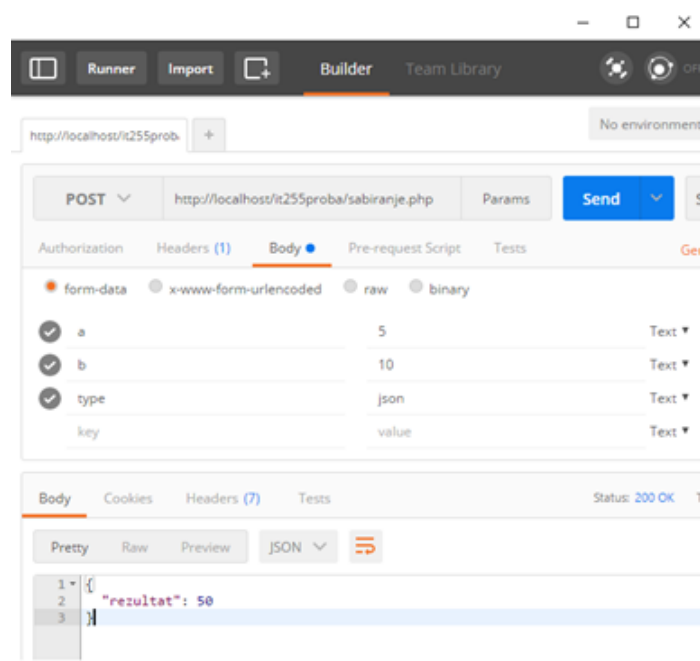
Ukoliko bi želeli da napravimo web servis koji može da množi dva broja moguće je da to uradimo preko POST servisa na sledeći način:

```
<?php  
$type = $_POST['type'];  
$a = $_POST['a'];  
$b = $_POST['b'];  
if($type == "json"){  
    header("Content-type: application/json");  
    $test_array = array (  
        'rezultat' => ($a*$b),  
    );  
    echo json_encode($test_array);  
}else {  
    header("Content-type: text/xml");  
    $test_array = array (  
        ($a*$b) => 'rezultat',  
    );  
    $xml = new SimpleXMLElement('<root/>');  
    array_walk_recursive($test_array, array ($xml, 'addChild'));  
    print $xml->asXML();  
}  
  
?>
```

Ovaj servis možemo pozvati na sledeća dva načina:



Slika 9.1.1 Primer poziva za XML odgovor



Slika 9.1.2 Primer poziva za JSON odgovor

▼ 9.1 Zadaci za individualnu vežbu

ZADACI ZA INDIVIDUALAN RAD

Cilj ove sekcije je da studentu da zadatke za vežbanje

Zadatak 1:

Napraviti PHP servis koji će vraćati listu svih proizvođača automobila

Zadatak 2:

Pronaći na internetu kako se iz PHP-a šalje mejl preko Gmail-a i potom napraviti servis koji

će poslati mejl sa Vašeg gmail-a nekome. Servis treba da bude POST servis i treba da primi naslov, sadržaj i mejl kome se šalje.

✓ Poglavlje 10

Domaći zadatak 7

DOMAĆI ZADATAK BROJ 7

Cilj ovog domaćeg zadatka je da student provežba naučeno na vežbama

Napraviti POST I GET servis po izboru i prikazati njegov rad kroz POSTMAN. Servisi moraju da rade nešto korisno (da imaju argumente koji se obrađuju zarad nekog rezultata).

Domaći zadatak smestiti na GitHub sa imenom commit-a *IT255-DZXX-ime-prezime-brojindeksa* i poslati na mejl: vuk.vasic@metropolitan.ac.rs sa naslovom IT255 – DZ07.

▼ Zaključak

SERVERSKI SKRIPTOVI – PHP JEZIK

PHP je skript programski jezik opšte namene.

PHP je skript programski jezik opšte namene. Prvobitno je dizajniran kao jezik za kreiranje dinamičkih veb strana. Omogućava brzo procesiranje i učitavanje strana, jednostavan je za razumevanje i korišćenje i izvršava se na skoro svim operativnim sistemima. PHP se procesira od strane veb servera i generiše XHTML kod ili neki drugi izlaz koji veb čitačimogu da prepoznaju i interpretiraju.

Da biste mogli da kreirate interaktivne vebstrane i da izvršavate PHP programe, neophodno je da imate pristupnekom serveru koji podržava PHP. Međutim, često je u toku kreiranja nekog interaktivnog veb sajta pomoću PHP-a veoma nepraktično da svaki put kada napravite neku PHP stranu, morate da je prebacite na udaljeni server kako biste je istestirali. Zbog toga se preporučuje da PHP instalirate lokalno, na svoj računar.

Jedna od najčešćih primena serverskih skript jezika je obrada HTML formi.

Osnovna arhitektura sistema za pristup bazama podataka iz Web aplikacija sastoji se od Web browser-a, Web servera, mašine za izvršavanje skriptova i servera baze podataka. Pre nego što se uspostavi veza sa bazom podataka, potrebno je proveriti ispravnost podataka koje je korisnik uneo.

Da bi se ispravno obradili podaci koje korisnik šalje, neophodno je filtrirati upravljačke znakove (" , ' \ i znak NULL) iz tih podataka. PHP podržava više funkcija za formatiranje stringova radi unošenja u bazu podataka. Pre prikazivanja korisniku, podatke dobijene iz baze podataka takođe treba filtrirati, kako bi se iz njih izbacio znak \ unet prilikom filtriranja ulaznih podataka, i izbegao pogrešan rad baze zbog specijanih znakova u HTML-u.

PHP5 ima novu biblioteku funkcija za uspostavljanje veza sa bazama podataka, koja se zove mysqli (i od *improved*). Ova biblioteka podržava upotrebu i objektno-orijentisane i proceduralne sintakse. Broj istovremenih veza koje se mogu uspostaviti sa MySQL serverom je ograničen parametrom `max_connections` koji se podešava u datoteci `my.conf`.

LITERATURA ZA LEKCIJU 07

U izradi ove lekcije korišćena je navedena literatura.

Obavezna literatura:

1. Learning Web Design, Fourth Edition, by Jennifer Niederst Robbins, Copyright © 2012 Littlechair, Inc.
2. HTML5 and CSS3 Responsive Web Design Cookbook, Benjamin LaGrone, Copyright © 2013 Packt Publishing.

Dopunska literatura:

1. Luke Welling, Laura Thomson, PHP i MySQL: Razvoj aplikacija za Web, prevod Mikro knjiga, 2006.
2. A. Jevremović i M. Veinović, "INTERNET TEHNOLOGIJE", UNIVERZITET SINGIDUNUM, Beograd, 2013.

Veb lokacije:

1. . <https://www.uniqtechnologies.co.in/>
2. http://www.link-elearning.com/kurs-PHP-programiranje_289_
3. <http://www.w3schools.com/>
4. <https://www.packtpub.com/>

