



Principi softverskog inženjerstva

Programski jezik PHP 7 Osnovni kurs

Autori:
D.Drašković, T.Šekularac, J.Cincović

Desktop i veb aplikacije



- Aplikacija je bilo koji program koji je napravljen da bi bio pojednostavljen ili obavljen neki zadatak.
- Desktop aplikacija je instalirana na klijentskoj mašini.
- Desktop (klasična) aplikacija se pokreće izvršavanjem određenog izvršnog fajla (exe / jar).
- Veb aplikacija je instalirana na nekom udaljenom server i njoj se pristupa putem internet-a.
- Veb aplikacija se pokreće zahtevom odgovarajućeg URL-a u veb pregledaču,
koji predstavlja interfejs veb aplikacija.

Aplikacije i baze podataka



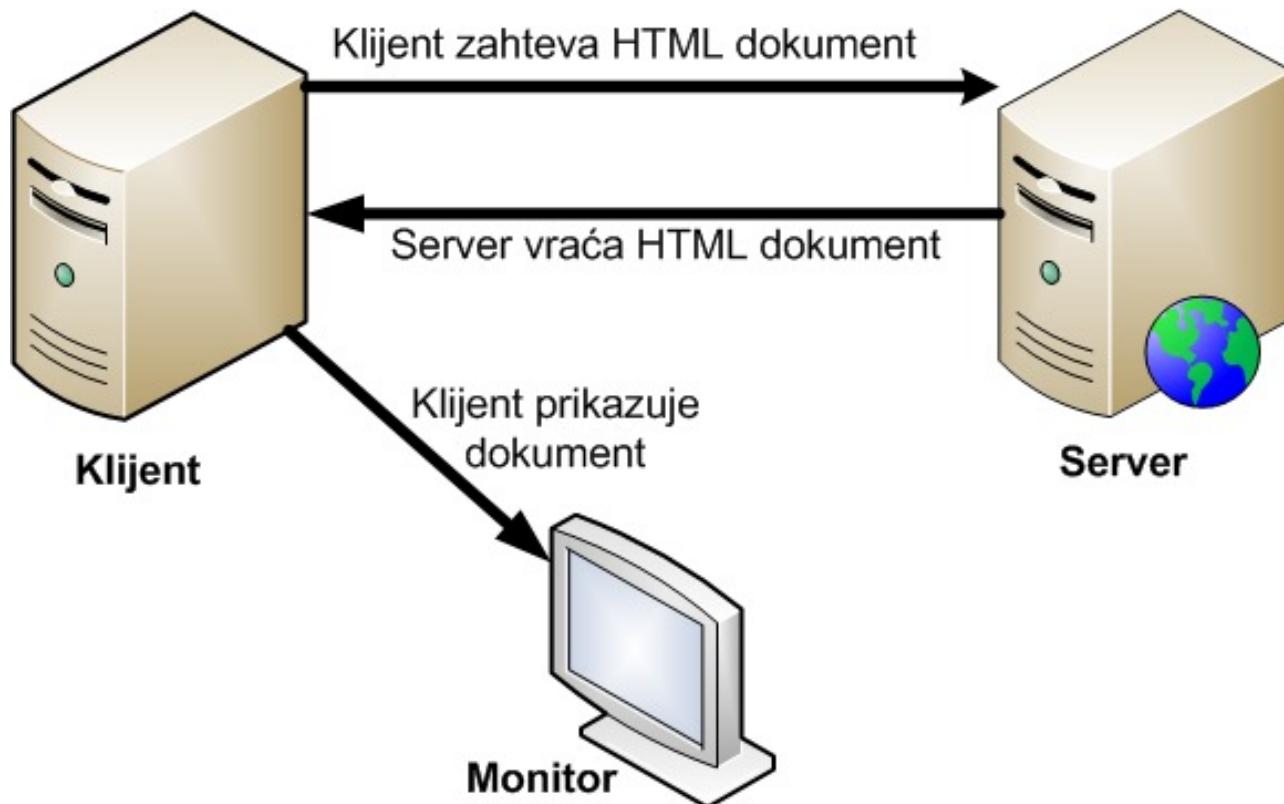
- Baza podataka se koriste za skladištenje podataka, ako klasična ili veb aplikacija rade sa većom količinom podataka.
- Podaci u bazi podataka su potpuno nezavisni od aplikacije.
- Posebnim naredbama aplikacija čita podatke iz baze podataka i ažurira podatke u bazi podataka (dodaje, briše, menja).

Statički veb sajтови



- Statički sajтови se ne menjaju sve dok sam autor nešto ne promeni
- Omogućavaju slanje informacija ka korisnicima
- Korisnici nemaju mogućnost interakcije i ne mogu neki zadatak da izvršavaju na programabilan način
- Statičke veb sajtove čine obične statičke HTML stranice (skup HTML fajlova)

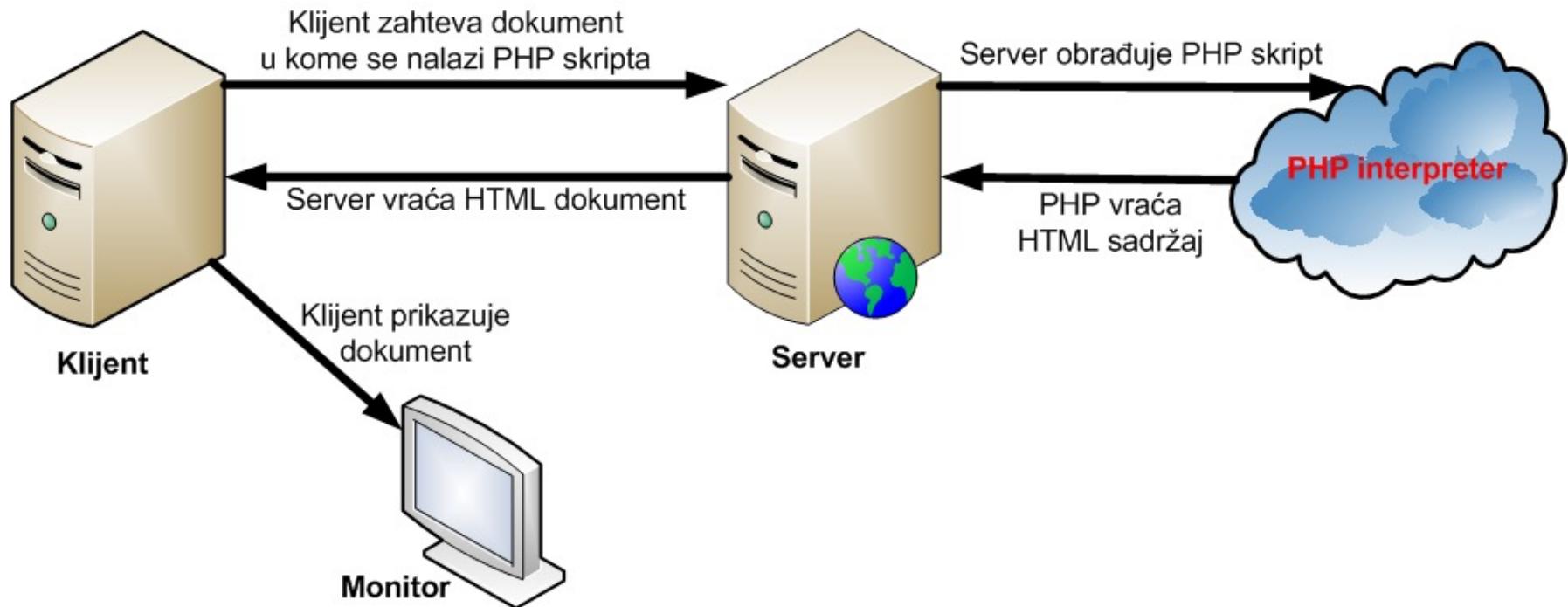
Zahtev za HTML dokumentom



Dinamičke veb aplikacije

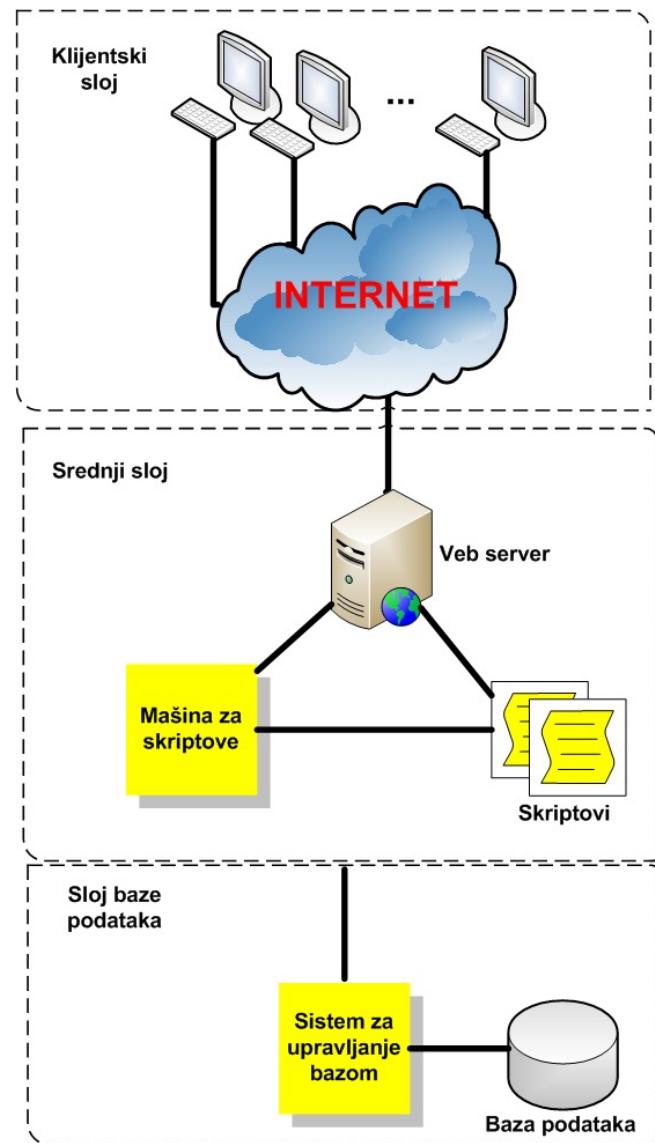


- Dinamičke veb aplikacije - interkacija sa korisnikom
- Dinamičke veb aplikacije čine dinamičke stranice (skup php, asp, jsp... fajlova) koje takođe koriste HTML jezik za komunikaciju sa klijentom
- Slanje zahteva kod klijent-server arhitekture:



Arhitektura troslojne veb aplikacije

- Većina veb aplikacija koje rade sa bazama podataka poseduje tzv. troslojnu arhitekturu koju čine sledeći slojevi:
 - klijentski sloj
 - aplikacioni sloj
 - sloj baze podataka



Kako su povezani slojevi?



- Sam veb obezbeđuje mrežu i protokole koji povezuju klijentski sloj sa srednjim slojem, a to je dominantno HTTP (HyperText Transfer Protocol).
- Komunikacija između srednjeg sloja i sloja baze podataka se obavlja pomoću protokola zavisnog od tipa programskog jezika i tipa baze podataka.
- Komunikacija preko HTTP-a dominira mrežnim saobraćajem na internetu (oko 80%).
- Za izradu veb aplikacija sa bazama podataka nije potrebno detaljno poznavati HTTP, ali je vrlo važno znati kakve probleme može da izazove HTTP u veb aplikacijama koje rade sa bazama podataka.

Osnove HTTP-a



- HTTP je standard koji omogućava prosleđivanje i deljenje dokumenata na veb-u.
- HTTP je jednostavan:
klijent (koji je u većini slučajeva veb pregledač) šalje HTTP serveru zahtev za određenim resursom, a server mu šalje svoj odgovor, u kome je sadržan i sam resurs - najčešće HTML dokument.

Čuvanje stanja



- Klasične aplikacije koje rade sa bazama podataka čuvaju stanje (korisnici se prvo prijavljuju, zatim izvršavaju pojedine transakcije, i na kraju kad obave posao odjavljuju se).
- HTTP ne čuva stanje tj. svaka interakcija između veb pregledača i veb servera je nezavisna od bilo koje druge interakcije.
- Dakle treba na neki način obezbititi čuvanje tekućeg stanja. Za čuvanje tekućeg stanja PHP obezbeđuje kolačiće i sesije.

Klijentski sloj



- Klijentski sloj u modelu troslojne arhitekture je veb pregledač (eng. *web browser*).
- Veb pregledač šalje HTTP zahteve za resursima, obrađuje HTTP odgovore i prikazuje HTML resurse.
- Prednosti upotrebe veb pregledača kao tankog klijenta:
 - jednostavan razvoj
 - nezavisnost od platforme na klijentskoj strani
- Najpopularniji veb pregledači:
Internet Explorer, Mozilla Firefox,
Google Chrome, Opera,...

Web pregledači (čitači)



- Zajedničke odlike pregledača (čitača) web-a:
 - Svi web pregledači su HTTP klijenti koji šalju zahteve i prikazuju odgovore dobijene od web servera (obično u nekom grafičkom okruženju)
 - Svi čitači tumače sadržaj HTML stranica
 - Neki čitači prikazuju slike, reprodukuju filmove i zvuk i vizuelizuju druge tipove objekata
 - Mnogi čitači mogu da izvršavaju JavaScript kod ugrađen u HTML stranice (npr. za proveru ispravnosti unetih podataka u HTML formi)
 - Nekoliko čitača može da primenjuje kaskadne liste stilova (Cascading Style Sheets, CSS)
- Postoje manje razlike u načinima na koji pojedini čitači web-a prikazuju HTML stranice. Npr. neki ne prikazuju slike ili ne izvršavaju Java Script kod itd.

URL



- URL-ovi (*Uniform Resource Locator*) koriste se na veb-u kao primarni način za imenovanje i adresiranje resursa.
- Drugim rečima, URL omogućava da se jedinstveno identificuje adresa nekog dokumenta/fajla na veb-u.

Struktura URL



- URL se može podeliti na tri osnovna dela:
 - identifikator protokola
 - identifikator ciljnog računara i servisa na njemu (port)
 - putanja (koja može da sadrži još i parametre)
- Struktura *URL*-a:
scheme://host:port/path?parameter=value#anchor
- Primer:
<http://en.wikipedia.org/w/wiki.phtml?title=Train&action=history>
- Sam HTTP standard ne ograničava dužinu *URL*-a,
ali to ipak čine neki stariji veb čitači i posrednički serveri.

Scheme - Identifikator protokola

- Prvi deo URL-a (scheme) služi da se identificuje aplikacioni protokol.
- Najčešće korišćeni protokoli:
 - http** - HyperText Transfer Protocol
(protokol za prenos hiperteksta)
 - https** - HyperText Transfer Protocol with SSL
(protokol za prenos hiperteksta putem veza zaštićenim SSL protokolom)
 - ftp** - File Transfer Protocol
(protokol za prenos fajlova)

Host - identifikator ciljnog računara

- Drugи deo URL-a (*host*) identifikuje ciljni računar (server).
- host** se može zadati preko imena ili kao IP adresa.
- Primeri:
 - www.w3.org
 - 18.29.1.35

Port - identifikator servisa na ciljnom računaru

- Treći deo URL-a (port) specificira TCP port number i služi za identifikaciju servisa na ciljnom računaru.
- Na internetu postoji konvencija da se dobro poznati priključak koristi na serveru koji pruža dobro poznati tip usluge (HTTP server očekuje zahteve na priključku 80, FTP server na priključku 21, itd).
- TCP priključak (port) nije fizički uređaj, već identifikator koji služi TCP softveru i omogućava uspostavljanje više virtuelnih veza sa istom mašinom za različite aplikacije.

Path - identifikator resursa na ciljnom računaru

- Četvrti deo URL-a (*path*) služi za identifikovanje resursa na ciljnom računaru, i to je putanja (najčešće stvarna putanja do ciljne datoteke na serveru).
- Ako se ime ciljne datoteke izostavi podrazumeva se index.html (ovo je po defaultu za Apache, mada se može i drugačije podesiti u fajlu httpd.conf).
- Primeri:
rfc/rfc1738.html
clanovi/novi_c/ ≡ clanovi/novi_c/index.html

Dodatni parametri u URL-u

- Peti deo URL-a (parameter=value) se koristi za prosleđivanje vrednosti parametara ciljnom resursu (npr. php fajlu) putem URL-a.
- U slučaju prosleđivanja više parova parametar - vrednost, isti se razdvajaju znakom &.
- Primeri GET parametara:

p1=Beograd

p1=Beograd&p2=2003

Anchor - identifikator fragmenta

- Šesti deo URL-a (*anchor*) predstavlja identifikator fragmenta se koristi za pozicioniranje na određeno mesto u okviru dokumenta.
- Ova mogućnost se koristi u slučaju velikih dokumenata.
- Mesto "skoka" mora biti registrovano u okviru dokumenta.

Aplikacioni sloj (#1)



- Aplikacioni sloj sadrži aplikacionu logiku koja se uglavnom realizuje preko skriptova .
- Skriptovi obavljaju sledeće funkcije:
 - Obavljaju određene zadatke karakterististične za svaku aplikaciju pojedinačno (izračunavanja,...) .
 - Komuniciraju sa DBMS-om na strani veb servera.
 - Generišu HTML kod potreban za prezentaciju podataka u korisnikovom veb pregledaču.

Aplikacioni sloj (#2)



- Najveći deo aplikacione logike nalazi se u aplikacionom sloju.
- Aplikacioni sloj obavlja većinu zadataka pomoću kojih se objedinjuju ostali slojevi:
 - Obrađuje podatke dobijene od korisnika pretvarajući ih u upite za čitanje ili upisivanje u bazu podataka.
 - Upravlja se strukturom i sadržajem podataka koji se prikazuju korisniku.
 - Omogućava upravljanje stanjem uz upotrebu HTTP protokola.
- PHP se nametnuo kao komponenta mnogih dinamičkih veb aplikacija srednjeg i velikog obima.

Osnovni pojmovi - PHP



Šta je PHP?

Jezik za pisanje skriptova koji rade na serveru,
namenski projektovan za upotrebu na vebu.

- Početna verzija PHP-a napravljena 1994. godine.
- Aktuelna verzija: PHP ver. 7.3 (od 7.3.2019.).
- Planirane buduće verzije: PHP 8 (2020) i PHP 9 (2025).
- PHP je proizvod otvorenog koda.
- Personal Home Page, **PHP Hypertext Preprocessor**
- <http://www.php.net>

Osnovni pojmovi - MySQL



Šta je MySQL?

Brz i robustan sistem za upravljanje
relacionim bazama podataka.

- MySQL je višekorisnički i višenitni sistem.
- SQL (*Structured Query Language*) je standardni
jezik za upite u bazi podataka.
- Zašto koristimo PHP i MySQL?

Mogu da rade pod svakim poznatijim operativnim
sistemom, čak i kod onih manje popularnih.

Prednosti PHP-a



- Visoke performanse
- Povezivanje s velikim brojem sistema za upravljanje bazama podataka
- Ugrađene biblioteke za obavljanje velikog broja poslova
- Niska cena
- Lako se uči i upotrebljava
- Dobra podrška za objektno orijentisano programiranje
- Prenosivost
- Izvorni kod dostupan svima
- Dobra podrška u slučaju problema
- Glavni konkurenti: Java, ASP.NET, Python, Ruby

Šta je novo u PHP verziji 7.3



- Bolja podrška za objektno orijentisano programiranje
- Deklaracija tipova povratnih vrednost funkcija
- Novi operatori: `<=>`, `??`
- Anonimne klase

Sloj baze podataka (#1)



- Sloj baze podataka se sastoji od sistema za upravljanje bazom podataka (*Database Management System - DBMS*) .
- DBMS omogućava:
 - Čitanje podataka iz baze
 - Ažuriranje podataka u bazi (unos, brisanje i izmena)
- CRUD - Create, Read, Update, Delete
- Organizacija prema objektima (entitetima) i odnosima koje postoje u sistemu na koji se baza podataka odnosi.
- Krajnji cilj integrisanosti je minimalna redundansa (višestruko pojavljivanje) podataka.

Sloj baze podataka (#2)



- Organizacija podataka prema potrebama korisnika

- Sigurnost:**

Podrazumeva efikasnu kontrolu pristupa podacima, u smislu ko može da pristupi bazi podataka, kojim podacima i šta može da radi sa tim podacima.

- Konkurentnost:** podrazumeva mogućnost sinhronizovanog rada više korisnika istovremeno.

- Integritet:**

Podrazumeva automatski oporavak od nasilnih prekida u toku rada koji dovode do tzv. nekonzistentnih stanja usled delimično izvršenih ažuriranja (unosa, izmene i brisanja) podataka.

Sloj baze podataka (#3)



- **Import:** Baza se često uvodi kao zamena za neki postojeći sistem, i tada mora postojati mogućnost preuzimanja tih podataka.
- **Eksport:** Javlja potreba da se postojeća baza zameni nekom još savremenijom bazom i tada postoji mogućnost predaje podataka bazi podataka na koju se prelazi.
- **Performanse:** Baza podataka mora da obezbedjuje maksimalni učinak u smislu najveće brzine rada uz najmanje zauzeće računarskih resursa.
- **Standardizacija:**
Standardni način opisa organizacije i operacija nad bazom obezbeđuje maksimalnu nezavisnost i trajnost korisničkog programa za rad sa bazom podataka u odnosu na promenu baze podataka.

Prednosti MySQL



- Visoke performanse

<http://web.mysql.com/benchmark.html>

- Niska cena

- Lako se konfiguriše i uči

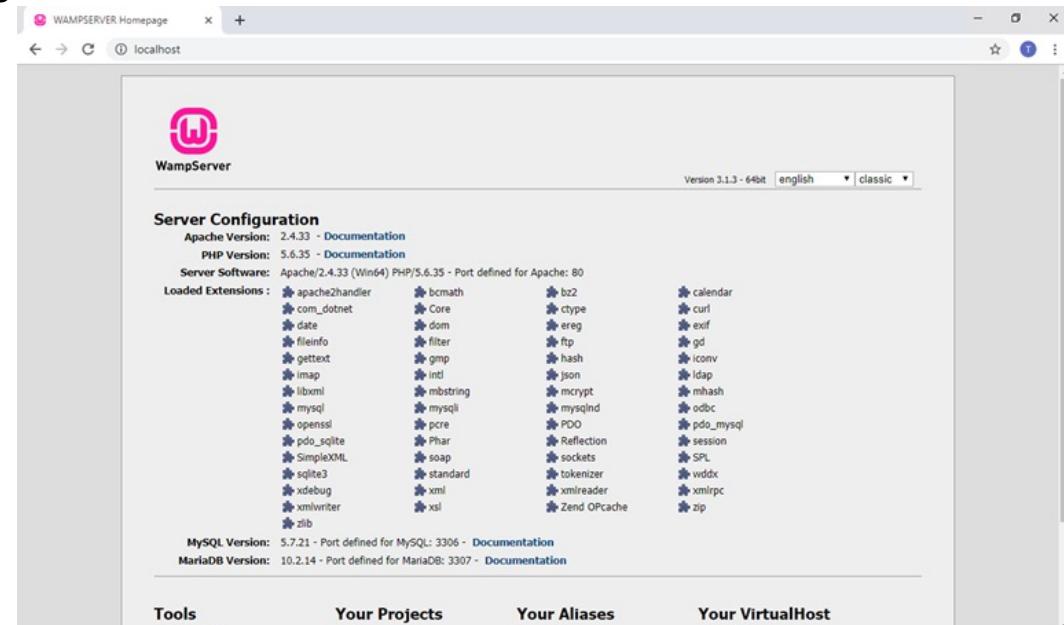
- Prenosivost

- Izvorni kod je javno dostupan

- Široko dostupna podrška u slučaju problema

Integrисane instalације - WAMP/XAMPP

- Integrисана verzija = WAMP ili XAMPP



- Razdvojena:

- Apache
- PHP
- MySQL

- Download Wamp: <http://www.wampserver.com/en/>
- Download XAMPP:
<https://www.apachefriends.org/download.html>
- Pristupa se preko: <http://localhost>

Razdvojena instalacija



- Instalacija Apache
- Podešavanje Apache servera
- Instalacija PHP
- Podešavanje PHP konfiguracije
- Instalacija MySQL

Okruženja za pisanje koda



- NetBeans

<https://netbeans.apache.org/>



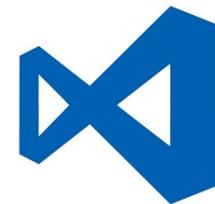
- PHPStorm

<https://www.jetbrains.com/phpstorm/download/>



- Visual Studio Code

<https://code.visualstudio.com/download>



- Eclipse:

<http://www.eclipse.org/downloads/>



- Aptana Studio

<http://aptana.com/downloads/start>



aptana®

1) Uvod u PHP



- PHP elementi
- Ugradnja PHP koda u HTML kod
- Promenljive i superglobalne promenljive
- Rad sa stringom i naredbe za štampanje
- Operatori
 - aritmetički, dodele, za nadovezivanje stringova, reference, operatori poređenja, logički operatori,...
- Uslovne strukture
 - if, if-else, switch
- Petlje
 - while, for, foreach, do... while

Primer 1 - HTML kod



• Neka je dat HTML kod:

```
<html>
<head>
    <title>Univerzitet u Beogradu</title>
</head>
<body>
    <h1>Elektrotehnički fakultet</h1>
    <h2>Odsek za softversko inženjerstvo</h2>
    <p>Neki tekst napisan u HTML kodu</p>
</body>
</html>
```

Primer 1 - PHP kod



- Sada ćemo ispod naslova, odnosno iza HTML taga </h2> da dodamo jedan PHP skript i da pokrenemo fajl u pregledaču veba:

```
<?php  
    echo '<p>Ovde ćemo da naucimo PHP</p>';  
?>
```

Primer 1 - Prikaz

A screenshot of a web browser window. The title bar says "Univerzitet u Beogradu". The address bar shows the URL "localhost:8080/Primeri/primer1.php". The main content area displays the following text:

Elektrotehnicki fakultet

Odsek za softversko inzenjerstvo

Ovde cemo da naucimo PHP

Neki tekst napisan u HTML kodu

Primer 1 - Izvorni kod



- Source code: PHP kod se ne vidi! Samo HTML...

The screenshot shows a Mozilla Firefox window with the title "Source of: http://localhost:8080/Primeri/primer1.php - Mozilla Firefox". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with "File", "Edit", "View", and "Help". The main content area displays the source code of a PHP file. The code is color-coded: tags like <html>, <head>, <title>, <body>, <h1>, <h2>, and <p> are in purple; text within these tags is in black. The code itself is:

```
<html>
  <head>
    <title>Univerzitet u Beogradu</title>
  </head>
  <body>
    <h1>Elektrotehnicki fakultet</h1>
    <h2>Odsek za softversko inzenjerstvo</h2>
    <p>Ovde cemo da naucimo PHP</p>      <p>Neki tekst napisan u HTML kodu</p>
  </body>
</html>
```

Ugradnja PHP koda u HTML kod

- PHP naredbe se ne vide, zato što je interpretator PHP koda zamjenio naredbe rezultatom.
- PHP se dakle dobija kao čisti HTML kod (zaključak: pregledač ne mora da razume PHP!)
- Skript jezici se razlikuju:
 - JavaScript = izvršava se kod klijenta,
 - PHP = izvršava se na veb serveru.

Elementi u PHP fajlu



U PHP fajlu mogu da postoje:

- HTML oznake (HTML tagovi),
 - PHP oznake (PHP tagovi),
 - PHP iskazi,
 - Beline i komentari.
-
- U primeru 1 najveći deo koda je HTML kod.

PHP oznake/tagovi



- Početni tag: <?php označava početak PHP koda
- Završni tag: ?> označava kraj PHP koda
- Ovi tagovi pokazuju veb serveru šta je PHP kod, i sve između <?php i ?> smatra se PHP kodom (izvan ovih oznaka veb pregledač smatra da je običan HTML kod)

Drugi PHP tagovi



- Postoje četiri stila pisanja PHP tagova:
 - XML stil (standardni)
`<?php echo 'ETF'; ?>`
 - Skraćeni stil (administratori nekad isključe korišćenje ove oznake - opcija short_tags)
`<? echo 'ETF'; ?>`
 - stil SCRIPT (slično kao JS)
`<SCRIPT LANGUAGE='php'> echo 'ETF'; </SCRIPT>`
 - ASP stil (samo u okruženjima ASP i ASP.NET, podrazumevano ovaj stil je isključen - opcija asp_tags)
`<% echo 'ETF'; %>`

PHP iskazi



- PHP iskazi (engl. *statements*) nalaze se između početnog i završnog taga i određuju šta interpretator PHP koda treba da uradi.
- U primeru 1, komandom echo štampamo (ispisujemo) jedan paragraf, i rezultat izvršavanja u veb pregledaču predstavlja tekst koji smo napisali u paragrafu.

Napomena:

Na kraju iskaza echo nalazi se znak ; (tačka zarez). Taj znak razdvaja izraze u PHP-u (kao u C-u ili Javi). Izostavljanje ; je česta sintaksna greška, koja se lako pravi, ali se lako i otkriva ☺

Beline



- Beline ili znakovi za razdvajanje:
novi red, znaci razmaka i tabulacija.
- Pregledači veba zanemaruju beline
i u HTML kodu i u PHP kodu.
- Primeri istog PHP koda:

```
echo '<h1>Dobrodosli u PHP</h1>';
i
echo '<h1>Dobrodosli      u      PHP      </h1> ';
echo 'Dobrodosli ';
echo 'na ETF! ';
i
echo 'Dobrodosli ';echo 'na ETF! ';
```

Komentari



- Komentari služe kao obaveštenje osobama koje čitaju programski kod.
- Šta se piše najčešće u komentarima?
- PHP podržava komentare slično kao C i C++
- Jednoredni komentar:

```
echo '<h1>Dobrodosli u PHP</h1>'; // komentar1  
echo '<h1>Dobrodosli u PHP</h1>'; # komentar2
```

- Višeredni komentar:

```
echo '<h1>Dobrodosli u PHP</h1>';  
/* Primer komentara  
   Drazen Draskovic  
   Izmenjen fajl: 01.03.2012.  
   Opis: Skript opisuje uvodnu lekciju.  
*/
```

Promenljive



- Svaka promenljiva (engl. *variable*) u PHP-u počinje znakom za dolar **\$**
- Nemojte da se nervirate, izostavljanje **\$** je uobičajena greška kod početnika na PHP-u ☺

String



- Predstavlja niz karaktera
- Deklaracije
 - apostrofi ' Primer1 '
 - navodnici "Primer2"
 - pomoćnih dokumenata <<< When_Will_It_End
- Bilo koja promenljiva se unutar stringa sa navodnicima pretvara se u string i izvršava,
dok se kod apostrofa ništa ne dešava!

```
$a=17;  
echo "Presli smo $a slajdova"; //stampa se 17  
echo 'Presli smo $a slajdova'; //stampa se $a
```

Štampanje - echo, print



- Mogu se koristiti:

```
echo 'zdravo';
```

```
echo 'zdravo', 'pozdrav';
```

```
echo ('zdravo');
```

```
print 'zdravo';
```

```
print ('zdravo');
```

Heredoc i Nowdoc sintaksa (<<<)

- Od PHP4 dodata je *heredoc* sintaksa, za korisnike Perl-a.
- Heredoc* sintaksa omogućava zadavanje dužih znakovnih podataka u lako razumljivom obliku, tako što se zadaje i graničnik kraja.
- Primer:

```
echo <<<kraj
    prvi red
    drugi red
    treći red
kraj
```
- Graničnik može biti bilo šta,
bitno je samo da se ne pojavljuje u tekstu!
- Nowdoc*, slično kao *Heredoc*, samo sa jednim navodnikom

Identifikatori



- Identifikatori su imena promenjivih, ali i imena funkcija i klasa.
- Identifikatori mogu da budu bilo koje dužine i mogu da se sastoje od slova, brojeva i _
- Identifikatori ne mogu da počinju cifrom.
- U PHP-u pravi se razlika između malih i velikih slova u imenima identifikatora (\$pr i \$Pr nije isto).
- Izuzetak:
Imena funkcija mogu se pisati i malim i velikim slovima!!!

Deklaracije



- U PHP-u ne treba da se deklariše promenljiva, pre nego što se upotrebi.
- Promenljiva će biti napravljena kada joj prvi put dodelimo vrednost:
`$ime = "Jasna";`
- Redefinisanje – ako preko iste promenljive, prepisemo novu vrednost:
`$ime = "Vesna"; //stara vrednost ne postoji`

Tipovi promenljivih



- *integer* - celobrojni tip, koristi se za cele brojeve
- *double* ili *float* - pokretni zarez dvostrukе preciznosti, koristi se za realne brojeve
- *string* - znakovni tip, koristi se za znakovne podatke
- *boolean* - logički, koristi se za podatke tipa tačno ili netačno
- *array* - niz, koristi se za čuvanje više podataka istog tipa
- *object* - objekat, koristi se za čuvanje primerka (instance) jedne klase

Specijalni tipovi podataka



- ➊ Dva specijalna tipa:
 - ➌ NULL - promenljive kojima nije dodeljena vrednost, koje su nedefinisane ili kojima je izričito dodeljena vrednost NULL.
 - ➌ Resurs - neke ugrađene funkcije (npr. za rad sa bazama podataka) vraćaju promenljive tipa resurs, koje predstavljaju spoljne resurse (veza sa bazom podataka).

Provera tipa podataka



- PHP je jezik sa slabom proverom tipa podataka.
- Tip podataka se ne navodi eksplicitno kod promenljivih u PHP-u, već se određuje na osnovu vrednosti koja joj je dodeljena:

```
$ime = "Jasna";           //ime je string  
$broj = 10;                //broj je integer  
$broj = "Desetka";        //broj je sada string
```

Konverzija tipa podataka



- Operator za konverziju omogućava privremenu promenu tipa promenljive ili vrednosti.
Postupak se odvija identično kao u jeziku C.
- Primer:

```
$broj = 10;           //integer  
$realbroj = (double) $broj; //double
```

Promenljiva promenljive



- PHP podržava tip - promenljiva promenljive (engl. *variable variable*).
- Takve promenljive omogućavaju da ime promenljive menjamo dinamički.
- Primer:

```
$ocena = "Deset";  
$ocena = 10;
```

što je ekvivalentno izrazu:

```
$Deset = 10;
```

Deklarisanje i upotreba konstanti

- Konstanta predstavlja određenu vrednost, isto kao i promenljiva, ali se ta vrednost zadaje jednom i potom se više ne može menjati.
- Primer:

```
define ('BEOGRAD', 11000);
```
- Kako razlikujemo konstante i promenljive?
 - Imena konstanti pišu se velikim slovom! (kao u C)
 - Ispred konstante se ne piše simbol dolar **\$**
- Upotreba konstante - samo se pozove njeno ime:

```
echo BEOGRAD; //izlaz: 11000
```
- Vrednost unutar define može biti i niz.

Skalarni tip deklaracije u PHP 7

- PHP 7 uvodi skalarne tipove (*Scalar*)
- Deklaracija skalarnog tipa označava da funkcija može prihvati parametre, određenog zadatog tipa:
 - tekstualni podaci (string),
 - celi brojevi (int),
 - realni brojevi (float),
 - bulove promenljive (bool)
- Dodavanjem skalarnih tipova, omogućeni su strogi zahtevi, pruža se veća kontrola nad kodom i programski kod postaje čitljivi
- Primer:

```
function FunctionName (int $a, int $b):int {  
    return $a + $b;  
}
```

Primer bez striktnih tipova



- Ukoliko su striktni tipovi zabranjeni (*disabled*) tzv. *Coercive mode*
- Primer:

```
<?php
    function dodavanje(float $a, float $b) : int
    {
        return $a + $b;
    }

    echo dodavanje(1,'2');
/*
 * bez striktnih tipova, PHP ce promeniti int(1) u float(1.0)
 * i string('2') u float(2.0) i povratni tip je strogo int(3)
 */
?>
```

Primer sa striktnim tipovima

- Ukoliko su striktni tipovi dozvoljeni (*enabled*)

- Primer:



// tzv. *Strict mode*

```
<?php
declare(strict_types=1);

function dodavanje(float $a, float $b) : int
{
    return (string) $a + $b;
}
echo dodavanje([], '2');
// Fatal error: Uncaught TypeError: Argument 1 passed to
// dodavanje() must be of the type float, array given
echo dodavanje(1.0, '2');
// Fatal error: Uncaught TypeError: Argument 2 passed
// to dodavanje() must be of the type float, string given
// Integers can be passed as float-points :
echo dodavanje(1,1);
// Fatal error: Uncaught TypeError: Return value of dodavanje()
// must be of the type integer, string returned
```

Opseg važenja promenljive (1)

- Izraz opseg važenja (engl. *scope*) odnosi se na mesta u skriptu na kojima je data promenljiva vidljiva.
- U PHP-u postoji 6 mogućih opsega važenja:
 - Ugrađene globalne promenljive
 - vidljive u celom skriptu.
 - Konstante imaju globalnu vidljivost,
 - pa se upotrebljavaju i unutar i izvan funkcija.
 - Globalne promenljive, deklarisane u skriptu, vidljive su svuda u skriptu, ali ne i unutar funkcija.

Opseg važenja promenljive (2)

(nastavak)

- Promenljiva koju napravite unutar funkcije, a ima isto ime kao promenljiva koja je deklarisana kao globalna, upućuje na globalnu promenljivu istog imena.
- Promenljive koje napravite unutar funkcije, a deklarišete kao statičke promenljive, ne vide se izvan funkcija, ali čuvaju svoju vrednost između dva izvršavanja funkcije.
- Promenljive koje napravite unutar funkcije lokalne su za funkciju i prestaju da postoje kada prestane izvršavanje funkcije.

Opseg važenja promenljive (3)

- Od PHP-a 4.1 nadalje, za nizove `$_GET` i `$_POST`, i za neke druge posebne promenljive, važe drugačija pravila za opseg važenja.
- To su superglobalne promenljive, koje se svuda vide, i u funkcijama i van njih.

Superglobalne promenljive



- **\$GLOBALS** - niz svih globalnih promenljivih
- **\$_SERVER** - niz serverskih promenljivih
- **\$_GET** - niz promenljivih prosleđenih skriptu metodom GET
- **\$_POST** - niz promenljivih prosleđenih skriptu metodom POST
- **\$_COOKIE** - niz kolačića
- **\$_FILES** - niz promenljivih koje se odnose na fajlove poslate sa klijentskog računara
- **\$_ENV** - niz promenljivih okruženja
- **\$_REQUEST** - niz svih promenljivih koje je korisnik poslao
- **\$_SESSION** - niz promenljivih sesije

Aritmetički operatori jezika PHP

- Aritmetički operatori su znakovi za računske operacije.

Operator	Ime	Primer
+	sabiranje	\$a + \$b
-	oduzimanje	\$a - \$b
*	množenje	\$a * \$b
/	deljenje	\$a / \$b
%	modulo (mod)	\$a % \$b

Primer

```
$a = 16;
```

```
$b = 10;
```

```
$rezultat = $a + $b; // rezultat je 26
```

```
$rezultat = $a % $b; // rezultat je 6
```

Operatori nadovezivanja



- Operator nadovezivanja znakovnih vrednosti (.) može se koristiti za spajanje dve ili više znakovnih vrednosti (slično kao sabiranje dva ili više brojeva).

Primer

```
$a = "Drazen ";
$b = 'casove';
$rezultat = $a." drzi ".$b;
```

Promenljiva rezultat će u ovom slučaju sadržati znakovnu vrednost: "Drazen drzi casove".

Operatori dodele



- Osnovni operator dodele (=)
- Čita se kao “dobija vrednost”
- Na primer: `$ukupno=3;` //ukupno dobija vred. 3
- Za sve operatore dodele važi sledeće pravilo:
vrednost celog iskaza dodele je vrednost koja je dodeljena
operandu na levoj strani.
- Na primer: `$b=6+($a = 5);` //b dobija vred. 11
- Zgrade se upotrebljavaju da se poveća prioritet
izračuvanja podizraza. Princip je isti kao u matematici!!!

Kombinovani operatori dodele

Operator	Upotreba	Ekvivalentan izrazu
<code>+=</code>	<code>\$a += \$b</code>	<code>\$a = \$a + \$b</code>
<code>-=</code>	<code>\$a -= \$b</code>	<code>\$a = \$a - \$b</code>
<code>*=</code>	<code>\$a *= \$b</code>	<code>\$a = \$a * \$b</code>
<code>/=</code>	<code>\$a /= \$b</code>	<code>\$a = \$a / \$b</code>
<code>%=</code>	<code>\$a %= \$b</code>	<code>\$a = \$a % \$b</code>
<code>.=</code>	<code>\$a .= \$b</code>	<code>\$a = \$a . \$b</code>

Kombinovani operatori dodele postoje za svaki aritmetički operator i za operator nadovezivanja znakovnih vrednosti

Operatori ++ i --



- Prefiksno uvećanje ++ i umanjenje --

- Primer (prefiksno uvećanje)

```
$a = 4;
```

```
echo ++$a;
```

```
// rezultat je 5, a=5
```

- Sufiksno uvećanje ++ i umanjenje --

- Primer (sufiksno uvećanje)

```
$a = 4;
```

```
echo $a++;
```

```
// rezultat je 4, a=5
```

- Slično se ponašaju i operatori umanjenja

Reference



- Operator referenca (&, ampersand) se može koristiti u kombinaciji sa operatorima dodele.

- Primer:

```
$a = 5;  
$b = $a;  
$a = 7; // b je i dalje 5  
-----  
$a = 5;  
$b = &$a;  
$a = 7; // i a i b su sada 7
```

- Poništavanje definicije jedne od promenljivih
`unset ($a);` Promenljiva b ne menja vrednost,
ali prekida se veza između \$a i 7.

Operatori poređenja



Operator	Ime	Upotreba
<code>==</code>	jednako	<code>\$a == \$b</code>
<code>====</code>	identično	<code>\$a === \$b</code>
<code>!=</code>	različito	<code>\$a != \$b</code>
<code>!==</code>	nije identično	<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>	manje od ili jednako	<code>\$a <= \$b</code>
<code>>=</code>	veće od ili jednako	<code>\$a >= \$b</code>
<code><=></code>	svemirski brod (<i>Spaceship</i>)	<code>\$a <=> \$b</code>

Komparacija različitih tipova

Operand 1	Operand 2	Rezultat
null ili string	string	konvertuje null u prazan string ""
bool ili null	bilo šta	konvertuje obe strane u bool (FALSE < TRUE)
object	object	različite klase su neuporedive, ugrađene klase mogu definisati sopstveno poređenje
string, resource, int or float	string, resource, int or float	translira string i resource tip u brojeve
array	array	niz sa manjim brojem članova je manji, ako ključ iz Operand_1 nije pronađen u Operand_2, onda su nizovi neuporedivi, u suprtonom – upoređuje se po vrednosti
object	bilo šta	objekat je uvek veći
array	bilo šta	array je uvek veći

Logički operatori



Operator	Ime	Upotreba	Rezultat
!	negacija	<code>!\$b</code>	Vraća <i>true</i> ako je <code>\$b</code> <i>false</i> i obrnuto
<code>&&</code>	konjunkcija	<code>\$a && \$b</code>	Vraća <i>true</i> ako su <code>\$a</code> i <code>\$b</code> <i>true</i>
<code> </code>	disjunkcija	<code>\$a \$b</code>	Vraća <i>true</i> ako su <code>\$a</code> i <code>\$b</code> ili oba <i>true</i>
and	konjunkcija	<code>\$a and \$b</code>	Isto kao <code>&&</code> , ali nižeg prioriteta
or	disjunkcija	<code>\$a or \$b</code>	Isto kao <code> </code> , ali nižeg prioriteta

Na primer koristimo pri utvrđivanju da li je vrednost promenljive `$a` između 0 i 100. To omogućava operator logičke konjunkcije (AND) :

`$a >= 0 && $a <= 100`

Operatori nad bitovima



Operator	Ime	Upotreba	Rezultat
&	konjunkcija	$\$a \& \b	Bitovi koji su aktivni u $\$a$ i $\$b$
	disjunkcija	$\$a \b	Bitovi koji su aktivni u $\$a$ ili $\$b$
~	negacija	$\sim \$a$	Bitovi koji su aktivni u $\$a$ nisu u $\$b$ i obrnuto
^	isključiva disj.	$\$a ^ \b	Bitovi aktivni ili u $\$a$ ili u $\$b$, ali ne u oba
<<	pomeranje ulevo	$\$a << \b	Pomera bitove $\$a$ ulevo za $\$b$ mesta
>>	pomeranje udesno	$\$a >> \b	Pomera bitove $\$a$ udesno za $\$b$ mesta

Još neki operatori



- Uslovni operator (slično kao if-else)

```
uslov ? uslov_true : uslov_false
```

```
($ocena > 5 ? ' Polozio ' : ' Pao ' );
```

- Operator zanemarivanja greške

```
$a = @(27/0)
```

Bez operatora @, izvršno okruženje bi generisalo upozorenje “elite nulom”.

Ako upotrebite ovaj operator, greška se zanemaruje (ali kad se zanemaruju upozorenja o greškama, trebalo bi da napišete kod za obradu grešaka).

Još neki novi operatori (PHP 7)

- Coalesce operator
echo \$poruka ?? ”;

Operator ?? uzima prvu vrednost koja je definisana i nema vrednost null.

- Operator upoređivanja – svemirski brod (slično funkciji *strcmp*)
echo 2 <=> 1;

Operator <=> vraća -1 ako je prvi element manji od drugog, 0 ako su jednaki i 1 ako je prvi veći od drugog.

Operatori za rad s nizovima

Operator	Naziv operatora	Upotreba	Rezultat
+	unija	$\$a + \b	niz od svih elemenata \$a i \$b
==	jednako	$\$a == \b	<i>true</i> , ako imaju jednake elemente
====	identično	$\$a === \b	<i>true</i> , ako imaju jednake elemente u jednakom redosledu
!=	različito	$\$a != \b	<i>true</i> , ako je \$a različit od \$b
<>	različito	$\$a <> \b	<i>true</i> , ako je \$a različit od \$b
!==	nije identično	$\$a !== \b	<i>true</i> , ako \$a nije identičan \$b
<=>	svemirski brod	$\$a <=> \b	0 (ako su isti), -1 (prvi manji od drugog), 1 (prvi veći od drugog)

Operator za utvrđivanje tipa - **instanceof** :

```
class nekaKlasa { };
nekiObjekat = new nekaKlasa();
if (nekiObjekat instanceof nekaKlasa) echo "nekiObjekat je primerak klase
nekaKlasa";
```

Prioriteti



- Svaki operator ima određeni prioritet, ili redosled kojim se izračunava.
- Svaki operator ima i asocijativnost (redosled izračuvanja za operatore jednakih prioriteta):
 - leva asocijativnost (sleva nadesno)
 - desna asocijativnost (zdesna nalevo)
 - nije bitna asocijativnost ($n \backslash b$)

Prioriteti - Tabela (1)



Asocijativnost	Operatori
leva	,
leva	or
leva	xor
leva	and
desna	print
leva	= += -= *= /= ,= %= &= = ^= ~= <<= >>=
leva	?:
desna	??
leva	
leva	&&
leva	
leva	^

najmanji
prioritet



nastavak tabele prioriteta na sledećem slajdu

Prioriteti - Tabela (2)



Asocijativnost	Operatori
leva	&
n\b	== != === !== <=>
n\b	< <= > >=
leva	<< >>
leva	+ - .
leva	* / %
desna	! ~ ++ -- (int) (double) (string) (array) (object) @
desna	[]
n\b	new
n\b	()



najveći
prioritet

Funkcije za rad s promenljivama

- **string gettype (promenljiva)**

- Utvrđuje tip promenljive koju joj prosledite i vraća znakovnu vrednost s imenom tipa (npr. boolean, integer, double, string array, resource, object ili NULL).

- **int settype (promenljiva, tip)**

- Menja tip promenljive koju joj prosledite u novi tip naveden u obliku znakovne vrednosti kao drugi argument.

Funkcije za proveru tipova podataka

- Funkcije za ispitivanje određenih tipova podataka:

- `is_array`
- `is_double`, `is_float`, `is_real` //ista f-ja
- `is_long`, `is_int`, `is_integer` //ista f-ja
- `is_string`
- `is_object`
- `is_resource()`
- `is_null()`
- `is_scalar()`
- `is_numeric()`
- `is_callable` /*Ispita da li vrednost promenljive ima postojeće funkcije.*/

Ispitivanje stanja promenljive

- **boolean isset (promenljiva)**
 - Vraća true, ako je promenljiva definisana, u suprotnom vraća false
- **void unset (promenljiva)**
 - Poništava definiciju promenljive koju joj prosledite
- **boolean empty (promenljiva)**
 - Ispituje da li postoji promenljiva i da li ima vrednost koja nije nula, odnosno nije prazna i vraća true ili false

Uslovne strukture - izraz if



- Ako je uslov ispunjen, izvršava se blok koda koji sledi iza if, u suprotnom se preskače

```
uslov
if($ocena == 5)
echo 'Pali ste ispit!<br/>';
//ako je uslov true biće izvršen
//izraz echo
```

Blok naredbi



- Često unutar jednog uslovnog iskaza, npr if, treba da se izvrši više iskaza:

```
if ($ocena > 5)  
{  
    echo '<font color=red>';  
    echo 'Polozili ste ispit!<br/>';  
    echo '</font>';  
}
```

blok

Uvlačenje koda



• Napomena:

Kao što smo naučili PHP zanemaruje praznine u kodu.

Radi bolje čitljivosti, trebalo bi da uvlačite iskaze. Uvlačenje se obično primenjuje da biste lakše uočili redove koji će biti izvršeni ako su ispunjeni uslovi, iskaze koji su grupisani u blokove i iskaze koji su deo petlji ili funkcija.

Iskazi else, elseif



- if (\$uslov == 0) {
...
} else {
...
}
• if (\$kolicina < 10)
 \$popust = 0;
elseif (\$kolicina >=10 && \$kolicina<=99)
 \$popust = 10;
elseif (\$kolicina >100)
 \$popust = 20;

Reč elseif može da se kuca i ovako i sa razmakom (else if), oba oblika su ispravna!

Iskaz switch



- Slično kao if, osim što kod switch, uslov može imati više različitih vrednosti, koje moraju biti skalarnog tipa (integer, string ili float).

```
switch ($navijac) {  
    case 'c': echo '<p>Crvena Zvezda</p>';  
                break;  
    case 'p': echo '<p>Partizan</p>';  
                break;  
    default:  echo '<p>Ne navija za klub</p>';  
                break;  
}
```

Petlje



- Ponavljanje nekih akcija (0 ili više puta)
- Petlja WHILE
- Petlje FOR i FOREACH
- Petlja DO .. WHILE

Petlja while



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

- Slično kao IF, samo se blok ne izvršava kad je uslov ispunjen, nego dokle god je uslov ispunjen.

```
$brojac=0;  
while ($brojac<=5) {  
    echo $brojac."<br/>";  
    $brojac++;  
}
```

Petlje for i foreach



```
for (izraz1; uslov; izraz2) izraz3;  
for ($i=1; $i<=$br_ljudi; $i++) {  
    $temp="ime$i";  
}
```

- Petlja foreach se koristi za rad sa nizovima.

Petlja do...while



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

- uvek se izvrši najmanje jednom
(nezavisno od uslova)

```
$brojac=100;  
do {  
    echo $brojac."<br/>"; }  
while ($brojac < 1);
```

Izlazak iz upravljačke strukture

- Postoje tri načina da se prekine izvršavanje bloka koda:

- break - iskakanje iz petlje
- continue - iskakanje na novu iteraciju petlje
- exit - prekidanje izvršavanja celog PHP skripta

```
if ( $broj_ispita == 0 )
{
    echo 'Nemate ispite za prijavljivanje!';
    exit;
}
```

Alternativni oblici sintakse



- Za sve navedene upravljačke strukture postoji alternativni oblik sintakse:
 - Početna vitičasta zagrada ({) zamjenjuje se dvotačkom (:)
 - Završna vitičasta zagrada (}) zamjenjuje se novom ključnom rečju, koja će biti endif, endswitch, endwhile, endfor ili endforeach, u zavisnosti koja upravljačka struktura je u pitanju.

```
if ( $stanje == 0) {  
    echo "Nemate dovoljno novca.";  
    exit;  
}
```

```
if ( $stanje == 0) :  
    echo "Nemate dovoljno novca.";  
    exit;  
endif;
```

Primer 2



- Napraviti narudžbenicu lekova u online apoteci, kao na slici:

Artikal	Kolicina
Andol	<input type="text"/>
Aspirin	<input type="text"/>
Vitamin C	<input type="text"/>

Kako ste saznali za nasu apoteku?

Ja sam redovan kupac ▾

Ja sam redovan kupac

TV reklama

Halo oglasi

Primer 2



- Kada kupac odredi količine i naruči lekove, kao potvrda treba da mu se odštampa fiskalni račun sa sledećim podacima:
 - a) datum i vreme kada su lekovi naručeni
(DINAMIČKI SADRŽAJ!!!)
 - b) ukupna količina naručenih lekova
 - c) količina po stavkama
 - d) ukupna cena računa bez poreza i sa porezom; stopa poreza je 8%;
 - e) “Hvala! Dođite nam ponovo!” ako nije redovni kupac

Primer 2 - Rezultat



Apoteka - narudzbina

Fiskalni racun

Roba narucena u 18:47, 29th April

Porucili ste:

Ukupna kolicina: 6

1 andol

2 aspirin

3 vitamin C

Ukupno bez poreza: 285.00 dinara

Ukupno sa porezom: 307.80 dinara

Hvala! Dodjite nam ponovo!

\$_POST, \$_GET i \$_REQUEST

- **\$_POST i \$_GET su superglobalni nizovi.**
- Jedan od nizova, ili **\$_POST ili \$_GET**, sadržaće vrednosti svih polja forme.
Koji niz će biti upotrebljen zavisi od metode POST/GET koja se koristi u formi:

```
<FORM NAME="forma" METHOD="POST" ACTION="prva.php">
<FORM NAME="forma" METHOD="GET" ACTION="prva.php">
```
- Svim poljima na **prva.php** se može pristupiti preko:
**\$_POST['naziv_polja'] ili
\$_GET['naziv_polja']**
- U nizu **\$_REQUEST** biće dostupni svi podaci, bilo da su poslati preko POST ili GET metode.

Rešenje (1)



```
<?php
    // kreiranje kracih imena varijabli
    $kolicina1 = $_POST['kol1'];
    $kolicina2 = $_POST['kol2'];
    $kolicina3 = $_POST['kol3'];
    $nadji = $_POST['nadji'];
?>
<html>
<head>
    <title>Online apoteka</title>
</head>
<body>
```

Rešenje (2)



```
<h1>Apoteka - narudzbina</h1>
```

```
<h2>Fiskalni racun</h2>
```

```
<?php
```

```
echo '<p>Roba narucena u ';
```

```
echo date('H:i, jS F');
```

```
echo '</p>';
```

```
echo '<p>Porucili ste: </p>';
```

```
$ukupno = 0;
```

```
$ukupno = $kolicina1 + $kolicina2 + $kolicina3; //sabiranje
```

```
echo 'Ukupna kolicina: '.$ukupno.'<br />';
```

Upotreba funkcije date()



- Funkcija date () očekuje kao argument znakovni niz, koji predstavlja format rezultata koji želite.
- Svako slovo u argumentu predstavlja jedan element datuma i vremena. Na primer:
 - H je oznaka za sat u 24-časovnom formatu
 - i je oznaka za minute, sa vodećom cifrom 0 za jednocifrene (0-9)
 - j je dan u mesecu, bez vodeće cifre 0
 - S predstavlja redni sufiks na engleskom (st/nd/rd/th)
 - F je pun naziv meseca

Rešenje (3)



```
if( $ukupno == 0) {  
    echo 'Niste kupili nista!<br />'; }  
else {  
    if ( $kolicina1>0 )  
        echo $kolicina1.' andol<br />';  
    if ( $kolicina2>0 )  
        echo $kolicina2.' aspirin<br />';  
    if ( $kolicina3>0 )  
        echo $kolicina3.' vitamin C<br />';  
    echo '<br />';  
}  
$ukupna_cena = 0.00;
```

Rešenje (4)



```
define('ANDOLCENA', 10);
define('ASPIRINCENA', 100);
define('VITCCENA', 25);

$ukupna_cena = $kolicina1 * ANDOLCENA      //mnozenje
            + $kolicina2 * ASPIRINCENA
            + $kolicina3 * VITCCENA;

echo 'Ukupno bez poreza: '.number_format($ukupna_cena,2).'
    dinara<br/>';

$porez = 0.08; // porez je 8%
$ukupna_cena = $ukupna_cena * (1 + $porez); //mnozenje
echo 'Ukupno sa porezom: '.number_format($ukupna_cena,2).'
    dinara<br/>';
```

Rešenje (5)



```
if ($nadji == 'a')
    echo '<p>HVALA!</p>';
else
    echo '<p>Hvala! Dodjite nam ponovo!</p>';
?
</body>
</html>
```

2) Upotreba nizova

- Nizovi sa numeričkim indeksima
- Nizovi sa drugačijim indeksima
- Operatori za rad sa nizovima
- Višedimenzionalni nizovi
- Sortiranje nizova
- Ostale radnje sa nizovima

Šta je niz?



- Niz (*engl. array*) je promenljiva određenog imena koja sadrži skup vrednosti u nekom redosledu
- Vrednosti smeštene u nizu nazivaju se *elementi*
- Jedan niz može imati više elemenata, a svaki element po jednu vrednost (tekst, broj, drugi niz)

Lista u primeru apoteke >

- Svakom elementu niza pridružuje se *indeks (ključ)*
- Višedimenzionalni niz je niz nizova
- PHP podržava numerički indeksirane i asocijativne nizove

Aspirin

Brufen

Vitamin C

Nizovi sa numeričkim indeksima

- PHP: indeksi uvek počinju sa nulom
- Inicijalizovanje: niz se pravi jezičkom konstrukcijom array ili []

```
$lekovi=array('Andol', 'Brufen', 'VitaminC');
```

```
$lekovi= ['Andol', 'Brufen', 'VitaminC'];
```

- Kopiranje jednog niza u drugi pomoću operatora =
- Funkcija range() automatski pravi niz rastućih brojeva

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

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

```
//2 je korak za koji se povećava
```

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

Pristupanje sadržaju niza (1)

- Sadržaju niza se pristupa pomoću imena promenljive i indeksa (ključa), koji se navodi u uglastim zagradama
- Primer:
\$lekovi [0], \$lekovi [1], \$lekovi [2]
- Isti sistem numerisanja kao u programskim jezicima C, C++, Java,... (indeks prvog u nizu je nula!)
- Sadržaj elementa niza menja se pomoću operatora =
\$lekovi [0] = 'Ampicilin';
Može se dodati i novi element \$lekovi [3] = 'Sirup' ;

Pristupanje sadržaju niza (2)

- Nizovi se automatski prave prvi put kada ih upotrebljavate
- Veličina se dinamički povećava kad god mu dodate nov element
- Prikazivanje sadržaja niza:

```
for ($i = 0; $i<3; $i++)  
    echo "$lekovi[$i] ";
```

- Za nizove se koristi i petlja foreach:

```
foreach ($lekovi as $tekuci_elem)  
    echo $tekuci_elem.' ';
```

```
foreach ($lekovi as $rb=>$vrednost)  
    echo "<p>$rb - $vrednost</p>";
```

Nizovi s drugačijim indeksima

- PHP podržava i nizove u kojima svakoj vrednosti možete da pridružite indeks koji želite
- Inicijalizovanje niza:

```
$lekovi=array ('aspirin'=>90, 'brufen'=>100,  
'vitaminc'=>25 );
```

Imena artikla su ključevi, a cene su vrednosti.

- Pristupanje elementima: \$lekovi ['aspirin']
- Može da se pravi i niz element po element:

```
$lekovi=['aspirin'=>90]; //pravi se niz  
$lekovi['brufen']=100;  
$lekovi['vitaminc']=25;
```

Operatori za rad sa nizovima

Operator	Ime	Primer
+	Unija	$\$a + \b
==	Jednako	$\$a == \b
====	Identično	$\$a === \b
!=	Različito	$\$a != \b
<>	Različito	$\$a <> \b
!==	Nije identično	$\$a !== \b



Operator unija pokušava da doda elemente niza \$b na kraj niza \$a.

Ako u nizu \$b postoje elementi koji imaju iste ključeve kao neki elementi iz niza \$a, ti ključevi se izostavljaju iz novog niza.

Višedimenzionalni nizovi



- Svaki element niza može da bude neki drugi niz
- Dvodimenzionalni nizovi = matrice

```
$matrica = array ( array ('sifral', 'aspirin', 90),  
                   array ('sifra2', 'brufen', 100),  
                   array ('sifra3', 'vitaminc', 25)  
 );
```

Sortiranje nizova - funkcija sort()

- Funkcija **sort()** daje niz sortiran po abecednom redosledu

- Primer:

```
$lekovi = array ( 'brufen', 'sirup', 'aspirin');  
sort($lekovi);
```

- Sortiranje može da se vrši i po numeričkom redosledu

- Drugi parametar funkcije sort je opcioni:

SORT_REGULAR (default), SORT_NUMERIC, SORT_STRING

Sortiranje nizova



- Funkcija **asort()** sortira niz prema vrednosti svakog elementa
- Funkcija **ksort()** sortira niz po ključu
- Primer:

```
$lekovi=array ('aspirin'=>90, 'brufen'=>100,  
               'vitaminc'=>25 );  
asort($lekovi); //po cenama: 25,90,100  
ksort($lekovi); //po kljucu tj abecedno
```

- Funkcije koje sortiraju niz po opadajućem redosledu:
`rsort()`, `arsort()`, `krsort()`

Promena redosleda elemenata niza

- Funkcija **shuffle()** nasumično menja redosled elemenata u nizu
- Funkcija **array_reverse()** pravi kopiju niza sa elementima u obrnutom redosledu
- Funkcija **array_push()** dodaje po jedan novi element na kraj niza
- Funkcija **array_pop()** uklanja poslednji element niza i vraća ga pozivajućem kodu

Pokazni primeri



```
$brojevi = array () ;  
for ($i=10; $i>0; $i--)  
array_push ($brojevi, $i) ;
```

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

Navigacija unutar nizova



- Kada napravimo novi niz, pokazivač upućuje na prvi element niza
- Rezultat `current($niz)` vraća tekući element niza
- Funkcija `next` pomera pokazivač
- Funkcija `next($niz)` prvo pomera pokazivač unapred, a zatim vraća nov tekući element
- Funkcija `reset($niz)` vraća pokazivač na prvi element u nizu (ili `false`, ako je niz prazan)
- Funkcija `end($niz)` pomera pokazivač na kraj niza
- Funkcija `prev($niz)` pomera pokazivač unatrag za jedno mesto, a zatim vraća nov tekući element

Prebrojavanje elemenata u nizu

- Funkcija `count()` vraća ukupan broj elemenata u nizu, koji joj je prosleđen
- Funkcija `sizeof()` radi isto što i funkcija `count()`
- Funkcija `array_count_values($niz)` broji koliko jedinstvenih vrednosti ima u nizu; funkcija vraća asocijativni niz, koji sadrži tabelu učestanosti (ključ je element niza, vrednost je broj ponavljanja)

Konstantni nizovi



- Do PHP 5.6 samo dodavanje klučne reči **const**
- Od PHP 7, može konstantni niz sa **define()**
- Primer:

```
<?php
define( 'JUNACI' , [
    'Paja',
    'Miki',
    'Silja'
]) ;
```

```
echo JUNACI[1]; //izlaz: "Miki"
?>
```

3) Rad sa znakovnim nizovima

- Formatiranje znakovnih vrednosti (stringa)
- Spajanje i razdvajanje znakovnih vrednosti
- Poređenje znakovnih vrednosti
- Regularni izrazi

Obavezna polja u formi



- Ukoliko postoje obavezna polja u formi, preporučljivo je da se to proveri pomoću funkcije isset()

Skraćivanje znakovnih vrednosti

- Funkcija `trim()` briše beline sa početka i kraja znakovnog podatka. Po definiciji briše znakove za povratak na početak reda (`\r`) i za prelazak u novi red (`\n`), horizontalne i vertikalne tabulatore (`\t, \x0B`), znakove za kraj znakovne vrednosti (`\0`) i razmake.
- Ovoj funkciji možete da prosledite i parametar koji sadrži eksplicitno zadat spisak znakova koje treba ukloniti umesto ovih navedenih.

Formatiranje znakovnih vrednosti u oblik pogodan za prikazivanje

- Funkcija `n12br()` prihvata znakovnu vrednost kao ulazni parametar i sve značke za prelazak u novi red u njoj zamjenjuje XHTML oznakama `
`, ili HTML oznakom `
` pre PHP4.
- Ovo funkcija je korisna za formatiranje teksta.
- Funkcija `printf()` prosleđuje formatiranu znakovnu vrednost pregledaču veba, a funkcija `sprintf()` vraća formatiranu znakovnu vrednost.

```
printf ("Ukupno prodato: %.2f dinara", $ukupno);  
// 2f označava broj sa dve decimale nakon zareza
```

Više specifikacija konverzije

- U istom šablonu formata možete zadati više specifikacija konverzije. Svaka specifikacija biće zamjenjena formatiranom vrednošću parametra, redosledom kojim su parametri navedeni.

```
printf ("Ukupan racun: %.3f dinara (sa porezom %.2f) ", $ukupno, $ukupno_pdv);
```

- Od verzije 4.0.6 moguće je numerisanje argumenata (tj. argumenti ne moraju da budu u istom redosledu kao specifikacije konverzije).

```
printf ("Ukupan racun: %2\$ .3f dinara (sa porezom %1\$ .2f) ", $ukupno_pdv, $ukupno);
```

Tipovi specifikacija konverzije

Tip	Značenje
b	Vrednost se tumači kao ceo broj, a prikazuje kao binarni broj
c	Vrednost se tumači kao ceo broj, a prikazuje kao znakovni podatak
d	Vrednost se tumači kao ceo broj, a prikazuje kao decimalni broj
f	Vrednost se tumači kao broj sa pokretnim zarezom dvostrukе preciznosti
o	Vrednost se tumači kao ceo broj, a prikazuje kao oktalni broj
s	Vrednost se tumači kao znakovna i prikazuje kao znakovna
u	Vrednost se tumači kao ceo broj, a prikazuje kao decimalni broj bez predznaka
x	Vrednost se tumači kao ceo broj, a prikazuje kao heksadecimalni broj sa malim slovima za cifre (a-f)
X	Vrednost se tumači kao ceo broj, a prikazuje kao heksadecimalni broj sa malim slovima za cifre (A-F)

Pretvaranje malih i velikih slova

- Funkcije za menjanje malih i velikih slova u znakovnim podacima

Upotreba	Opis	Vrednost
<code>\$ime</code>		Idete li na Eurosong?
<code>strtoupper(\$ime)</code>	sva slova menja u velika	IDETE LI NA EUROSONG?
<code>strtolower(\$ime)</code>	sva slova menja u mala	idete li na eurosong?
<code>ucfirst(\$ime)</code>	menja u veliko slovo prvi znak ulaznog argumenta, ako je alfabetSKI	Idete li na Eurosong?
<code>ucwords(\$ime)</code>	menja u veliko slovo prvi znak svake reči koja počinje alfabetSKIM znakom	Idete Li Na Eurosong?

Formatiranje znakovnih vrednosti u oblik pogodan za unošenje u bazu

- Određeni znakovi, koji su potpuno prihvativi u znakovnim vrednostima, mogu da izazovu probleme prilikom unošenja u bazu podataka, zato što baza može da ih protumači kao upravljačke znakove.
- Znakovi koji prave probleme:
 - navodnici (obični i polunavodnici)
 - obrnuta kosa crta (\)
 - znak NULL

Funkcija addslashes(\$string)

- \$tekst = addslashes (\$tekst) ;
- Pomoću ove funkcije treba formirati znakovni podatak, pre upisivanja u bazu.
- Argument ove funkcije je znakovna vrednost, a rezultat je formatirana znakovna vrednost.
- Primer:

```
$tekst = "Shaquille O'Neil";  
$tekst = addslashes ($tekst) ;  
//Rezultat: Shaquille O\\'Neil
```

Funkcija stripslashes(\$string)

- Ova funkcija uklanja kose crte, koje je postavila funkcija addslashes (\$string) ;

Magični navodnici



- Novije verzije PHP-a automatski uključuju direktivu `magic_quotes_gpc`. To znači da se automatski konvertuju sve znakovne promenljive iz GET, POST i COOKIE.
- Provera da li je direktiva uključena pomoću:
`get_magic_quotes_gpc()` pa ako vraća rezultat true, to znači da se znakovne promenljive iz GPC konvertuju automatski, pa ćete za prikazivanje podataka korisniku morati da obradite podatke pomoću funkcije `stripslashes($string)`.

Spajanje i razdvajanje znakovnih vr.

- explode(string granicnik, string text)
- Funkcija deli znakovnu vrednost text na mestima gde najde na znakovnu vrednost granicnik.
- Funkcija vraća niz.

Primer:

```
$email_array = explode ('@', $email);  
$email_array[0] = draskovic  
$email_array[1] = etf.bg.ac.rs
```

- Suprotne ovoj funkciji su implode() i join()

Funkcija substr()



- Ova funkcija omogućava izdvajanje dela ulazne znakovne vrednosti koja se nalazi između zadatog početnog i krajnjeg položaja.

- ```
string substr (string tekst, int pocetak
[, duzina]);
```

- **Primer:**

```
$test = "Danas je Dan republike!";
echo substr($test, 5); //je Dan republike!
echo substr($test, 0, 5); //Danas
```

# Poređenje znakovnih vrednosti

- `int strcmp(string str1, string str2)`  
Funkcija poredi dve znakovne vrednosti str1 i str2 i ako su jednake funkcija vraća 0. Ukoliko po abecednom redosledu vrednost str1 dolazi iza str2, funkcija vraća broj veći od nule, u suprotnom vraća broj manji od nule.
- `int strcasecmp(string str1, string str2)`  
Ista kao prethodna, samo ne pravi razliku između malih i velikih slova.

# Dužina znakovne vrednosti



- Dužina znakovne vrednosti (ukupan broj znakova) možete utvrditi pomoću funkcije strlen()

```
strlen('Hello') je 5
```

# Nalaženje stringova unutar drugih stringova



- Funkcija `strstr()` omogućava traženje date znakovne vrednosti ili znaka unutar druge znakovne vrednosti.
- Idenična ovoj funkciji je i funkcija `strchr()`
- Prototip funkcije `strstr()`:  
`string strstr (string tekst, string uzorak);`
- Funkcija `stristr()` je gotovo identična izvornoj funkciji, s tim što ne pravi razliku između malih i velikih slova u tekstu koji traži.

# Utvrđivanje položaja zadatog znakovnog podniza



- Funkcije `strpos()` i `strrpos()` su slične funkciji `strstr()`, ali ne vraćaju izdvojeni deo ulaznog paramtera, nego položaj parametra uzorak u parametru tekst.

- Prototip funkcije `strpos()`:

```
int strpos(string tekst, string uzorak, int [pomak]);
//pomak je mesto odakle pocinje pretraga u tekstu
```

- Primer:

```
$test = "Dobrodosli u Srbiju!";
echo strpos($test, 'o'); //rezultat: 1
echo strpos($test, 'do', 3); //rezultat: 5
```

- Funkcija `strpos()` radi brže od funkcije `strstr()`.

# Zamenjivanje delova znakovnog podatka (1)

- Funkcije `str_replace()` i `substr_replace()`
- Funkcionalnost je dobra za cenzurisanje pojedinih izraza, na primer na forumima.
- Prototip funkcije:

```
mixed str_replace (mixed uzorak, mixed zamena,
mixed tekst[, int &koliko_puta]);
```

- Rezultat funkcije je nova verzija prosleđenog teksta **tekst**, u kome je uzorak **zamenjen** vrednošću drugog parametra **zamena**.
- Opcioni parametar `koliko_puta`, kaže koliko puta treba da se izvrši zamena (tek od PHP 5).

# Zamenjivanje delova znakovnog podatka (2)

- Funkcija `substr_replace()` pronađe i zamjenjuje zadati podniz u zavisnosti od njegovog položaja.

- Prototip funkcije:

```
string substr_replace (string tekst, string
zamena, int početak, int [duzina]);
```

- Ova funkcija zamjenjuje deo parametra `tekst`, vrednošću drugog parametra `zamena`.
- Deo teksta u kome će biti izvršena zamena određen je trećim i opcionalno četvrtim parametrom.

# Primer 3



Napraviti formular za registrovanje korisnika na studentskom forumu koji će sadržati polja za ime, telefon, e-mail adresu i potvrdu e-mail adrese. Kada korisnik unese mail koji nije u formatu: *ime@domen* treba da se ispiše poruka da je pogrešan unos maila. Ako korisnik ne unese potvrdu maila treba da se ispiše da je pogrešno potvrđen mail. Kada korisnik potvrdi mail, ispisuje se poruka da će šifra korisniku *ime* na mail *ime@domen*.

## Otvaranje naloga za studentski forum

|                                             |                     |
|---------------------------------------------|---------------------|
| Ime i prezime                               | Drazen Draskovic    |
| Mobilni                                     | 069 234567          |
| Unesite e-mail adresu                       | drazen@etf.bg.ac.yu |
| Potvrdite vasu e-mail adresu                | drazen@             |
| <input type="button" value="Registruj me"/> |                     |

# Rešenje (1)



```
<?php
 // pravljenje kratkih imena
 $ime = $_POST['ime'];
 $mob = $_POST['mob'];
 $email1 = $_POST['email1'];
 $email2 = $_POST['email2'];

 $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
 //ovo mora da postoji

 $uporedi = strcmp($email1, $email2);
 $email_niz = explode('@', $email1);
 $brojac = count($email_niz);
```

# Rešenje (2)



```
if (strlen($email_niz[1])==0)
echo 'Niste lepo uneli e-mail adresu!!!
Pokusajte ponovo.';
else if ($uporedi!=0)
echo 'Niste potvrdili mail adresu,
pokusajte ponovo registraciju.';
else
echo 'Uspesna registracija naloga:
' . $email_niz[0] . '
 Sifra za forum
ce uskoro stici na vas mail: ' . $email2;
?>
```

# Primer 4



- Napravite formu sa checkbox-ovima gde korisnik može da između 4 muzičke grupe ili žanra odabere one koje sluša. Ono što sluša treba beležiti u niz, nepoznate veličine, a kada korisnik odabere i pritisne dugme potvrde potrebno je da se prikaže njegov izbor tj. da se isčita iz niza. U slučaju da korisnik ništa ne odabere, treba da se ispiše da ništa nije izabrano.

**Izaberite muziku koju slusate**

Bon Jovi  Police  Madonna  RHCP

# Rešenje (1)



```
<html>
<body>
<h1>Izaberite muziku koju slusate</h1>
<?php
 if (!isset($_POST['submit'])) {
?>
<form action="<?php echo $_SERVER['PHP_SELF'];
?>"method="POST">
 <input type="checkbox" name="muzika[]" value="Bon Jovi">Bon
Jovi
 <input type="checkbox" name="muzika[]" value="Police">Police
 <input type="checkbox" name="muzika[]" value="Madonna">Madonna
 <input type="checkbox" name="muzika[]" value="RHCP">RHCP
 <input type="submit" name="submit" value="Izaberi">
</form>
```

# Rešenje (2)



```
<?php
}
else {
 if (is_array($_POST['muzika'])) {
 echo 'Izabrali ste:
';
 foreach ($_POST['muzika'] as $nesto)
{
 echo "<i>$nesto</i>
";
}
 } else {
echo 'Nista nije izabrano';
}
?>
</body> </html>
```

# Funkcija eval()



- Funkcija eval() obrađuje prosleđeni znakovni niz kao PHP kod. Primer:

```
eval ("echo 'Zdravo svete';");
```

- Rezultat će biti isti kao da smo napisali:

```
echo 'Zdravo svete';
```

- Zašto je korisna ova funkcija?

- Ako u bazi čuvamo blokove koda koje ćemo potom učitavati i izvršavati pomoću funkcije eval()
  - Generisanje koda u petlji i izvršavanje pomoću funkcije eval()

# Funkcija eval() - Prednosti



- Funkcija eval() obično se koristi kao deo sistema šablonu.
- Iz baze podataka može da se učita mešavina HTML koda, PHP koda i običnog teksta.  
Vaš sistem šablonu može da formatira taj blok podataka i da pomoći funkcije eval() izvrši PHP kod iz njega.
- Funkcija eval() primenjuje se pri ažuriranju ili ispravljanju postojećeg koda.

# Komande die i exit



- Prekid izvršavanja skripta pomoću die i exit
- Komanda exit ne vraća ništa. Drugo ime za tu komandu je die. Ove komande mogu da se komanduju sa operatorom OR.
- Primer:

```
exit;
exit('Izvrsavanje skripta je prekinuto');
```

- Česte greške nastaju kada želite da uspostavite vezu sa bazom ili izvršite neki upit nad bazom. Dobro je da tada korisnika obavestite o grešci:

```
mysql_query($upit) or die("Upit ne moze da
se izvrsi" . mysql_error());
```

# Regularni izrazi



- Pomoću regulalnih izraza (regular expresions) može se utvrditi da li neki podaci tipa string zadovoljavaju određene složene šablone (datum, adresa e-pošte,...).
- Funkcije za rad sa regularnim izrazima omogućavaju čak i izdvajanje i zamenu složenijih podvrednosti unutar zadatog stringa.
- Iako funkcije za rad sa regularnim izrazima pružaju znatno više mogućnosti nego prethodno opisane funkcije, treba ih izbeći kad god je to moguće zato što troše dosta resursa (prostornih i vremenskih).

# POSIX-ova sintaksa za regularne izraze #1

- . - bilo koji znak (džokerski znak)
- \. - znak "." (tačka)
- \\ - znak "\" (obrnuta kosa crta)
- Da biste izbegli zabunu kada kao regularan izraz zadajete tačno određeni string bolje je da koristite jednostrukе navodnike
- [ ] - bilo koji od znakova zadatih unutar liste znakova navedenih između [ i ]; unutar liste se navode pojedinačni znaci i/ili opsezi znakova

# POSIX-ova sintaksa za regularne izraze #2

- [^ ] - nijedan od znakova zadatih unutar liste znakova navedenih između [^ i ]; unutar liste se navode pojedinačni znaci i/ili opsezi znakova
- Ako želite da se znak ^ tumači kao običan znak postavite ga na bilo koje mesto osim na prvo unutar srednjih zagrada.
- Ako želite da se znak - (znak za opseg) tumači kao običan znak stavite ga na početak ili na kraj liste unutar srednjih zagrada.

# POSIX-ova sintaksa za regularne izraze #3

- Pomoću sidra u regularnom izrazu može se zadati da uzorak treba da se nalazi na početku ili na kraju vrednosti koja se ispisuje.
- Znak ^ sidri uzorak na početak, a znak \$ na kraj znakovne vrednosti.
- Obe vrste sidra se koriste u regularnom izrazu u kome se zahteva potpuno poklapanje.

# POSIX-ova sintaksa za regularne izraze #4

- Znak ? znači 0 ili 1 pojavljivanje
- Znak \* znači 0,1 ili više pojavljivanja
- Znak + znači 1 ili više pojavljivanja
- Fiksni broj pojavljivanja tražene podvrednosti može se zadati između vitičastih zagrada, u obliku {min\_broj}
- Pomoću sintakse sa vitičastim zagradama mogu se zadati minimalan i maksimalan broj pojavljivanja, u obliku {min\_broj,max\_broj}

# POSIX-ova sintaksa za regularne izraze #5

- Znaci ?, \*, +, { } se mogu pojavljivati iza znaka, liste znakova ili podizraza regularnog izraza (razdvojenog zagradama) i u tom slučaju označava određeni broj ponavljanja znaka, liste znakova, podizraza regularnog izraza .

# POSIX-ova sintaksa za regularne izraze #6

- Podizrazi u regularnom izrazu se uokviruju malim zagradama.
- Mogućnost grupisanja izraza u podizraze omogućava definisanje složenih regularnih izraza i izdvajanje vrednosti koje odgovaraju podizrazima u regularnom izrazu u niz.
- Alternativni uzorci se razdvajaju pomoću operatora | koji ima najniži prioritet od svih operatora.

# POSIX-ova sintaksa za regularne izraze #7

- Za predstavljanje specijalnih znakova koristi se sekvenca obrnuta kosa crta - specijalni znak.
- Drugi način da se predstavi specijalan znak je da se on stavi u listu znakova ako tamo nema specijalno značenje.
- U regularnim izrazima mogu se upotrebljavati metaznaci (\t - tabulator, \n - novi red, \d - bilo koja cifra, \s - bilo koja belina).

# Funkcije za rad sa regularnim izrazima

- Pronalaženje i izdvajanje vrednosti:

```
int preg_match (string uzorak, string izvor
[, array &$matches [, int $flags = 0
[, int$offset = 0]]])
```

- Zamena delova izvorne vrednosti:

```
mixed preg_replace(string pattern,
 string replacement, string string
[, int $limit = -1 [, int &$count]])
```

- Razdvajanje izvorne vrednosti na elemente niza:

```
array preg_split (string $pattern ,
 string $subject [, int $limit = 1
 [, int $flags = 0]]])
```

# Funkcija preg\_match() (#1)



```
int preg_match (string uzorak, string izvor
[, array &$matches [, int $flags = 0
[, int $offset = 0]]]
```

- Proverava se da li se unutar stringa izvor nalazi deo koji odgovara regularnom izrazu uzorak (case sensitive).
- Ako se opcioni parametar *regs* ne koristi funkcija vraća vrednost 1 u slučaju da se unutar stringa *izvor* nalazi deo koji odgovara regularnom izrazu *uzorak*. U suprotnom vraća se false.

# Funkcija preg\_match() (#2)

- Parametar matches se koristi da se u njemu čuvaju delovi stringa *izvor* koji odgovaraju regularnom izrazu i delovima regularnog izraza (koji se u tom slučaju moraju nalaziti unutar malih zagrada).
- \$regs[1] će sadržati podstring parametra *izvor* koji odgovara delu regularnog izraza koji počinje od prve leve zgrade; \$regs[2] podstringu koji počinje od druge leve zgrade, itd.  
\$regs[0] će sadržati kopiju kompletног stringa koji odgovara regularnom izrazu.

# Regуларни изрази - Primer



```
$pattern = "/[_a-zA-Z0-9-]+(\.[_a-zA-Z0-9-]+)*@[a-zA-Z0-9-]+\(\.[a-zA-Z0-9-]+\)*(\.\[a-zA-Z\]{2,})/i";
$match_string="Mail: tasha@etf.rs";
if (preg_match($pattern, $match_string,$email)) {
 echo $email[0];
} else {
 echo "Mail nije pronađen";
}
```

## 4) Rad sa datumom i vremenom

- Funkcija date () ima dva parametra:
  - vrednost znakovnog tipa, koja predstavlja zahtevani format rezultata
  - vrednost u formatu Unixove vremenske oznake
- Tipičan primer funkcije date ():  
echo date ('d.m.Y.') ;
- Rezultat će biti datum u obliku: 22.03.2012.
- Neki šabloni za formate navedeni su u sledećoj tabeli.

# Šabloni za date() funkciju (1)

Kod	Opis
a	Prikazuje <b>am</b> ili <b>pm</b> , u zavisnosti od doba dana
A	Prikazuje <b>AM</b> ili <b>PM</b> , u zavisnosti od doba dana
c	Datum u formatu ISO8601: <b>GGGG-MM-DD-TČČ:MM:SS+GMTtime</b> Primer: 2012-03-22-T22:45-21:45
d	Dan u mesecu datuma, kao dvocifren broj sa vodećom nulom. Opseg: <b>01-31</b>
D	Dan u nedelji kao skraćenica od 3 slova. Opseg: <b>Mon-Sun</b>
F	Puno englesko ime meseca. Opseg: <b>January-December</b>
g	Čas u 12-časovnom formatu, bez vodeće nule. Opseg: <b>1-12</b>
G	Čas u 24-časovnom formatu, bez vodeće nule. Opseg: <b>0-23</b>
h	Čas u 12-časovnom formatu, sa vodećom nulom. Opseg: <b>01-12</b>
H	Čas u 24-časovnom formatu, sa vodećom nulom. Opseg: <b>00-23</b>
i	Minuti u času sa vodećom nulom. Opseg: <b>00-59</b>
I	Režim računanja vremena. <b>1 za letnje računanje vremena, 0 za zimsko.</b>

# Šabloni za date() funkciju (2)

Kod	Opis
j	Dan u mesecu bez vodeće nule. Opseg: <b>1-31</b>
I	Puno (englesko) ime dana u nedelji. Opseg: <b>Monday-Sunday</b>
L	Podatak o tome da li je godina prestupna. Vraća vrednost <b>1 za datum u prestupnoj godini, a 0 ako datum nije u prestupnoj godini.</b>
m	Mesec u godini kao dvocifren broj sa vodećom nulom. Opseg: <b>01-12</b>
M	Mesec u godini kao skraćenica od 3 slova. Opseg: <b>Jan-Dec</b>
n	Mesec u godini kao broj bez vodeće nule. Opseg: <b>1-12</b>
O	Razlika u časovima između vremena u tekućoj zoni i GMT.
r	Datum i vreme u formatu RFC822. Primer: <b>Sun, 15 Apr 2012 11:55:30 +0100</b>
s	Sekunde u minuti sa vodećom nulom. Opseg: <b>00-59</b>
S	Sufiks (engleski) koji opisuje redni broj dana. Iz skupa: <b>st, nd, rd, th</b>
t	Ukupan broj dana u mesecu. Opseg: <b>28-31</b>
T	Vremenska zona servera u obliku skraćenice od 3 znaka. Primer: EST

# Šabloni za date() funkciju (3)

Kod	Opis
U	Ukupan broj sekundi koji je protekao od 1.januara 1970.godine do tekućeg trenutka. Ovo je poznat Unixov format vremenske oznake.
w	Redni broj dana u sedmici kao jednociifren broj. Opseg: <b>0 (nedelja) do 6 (subota)</b>
W	Redni broj sedmice u godini, prema standardu ISO-8601.
y	Godina kao dvocifren broj. Na primer: <b>12</b>
Y	Godina kao četvorocifren broj. Na primer: <b>2012</b>
z	Redni broj dana u godini kao broj. Opseg: <b>0-365</b>
Z	Pomak tekuće vremenske zone, u sekundama. Opseg: od -43200 do 43200

# Unix-ove vremenske oznake

- Vreme u formatu Unix-ove vremenske oznake predstavlja tzv. vremensku marku (engl. *timestamp*)
- Na većini sistema koji rade pod Unix-om tekući datum i vreme čuvaju se u obliku 32-celobrojne vrednosti koja predstavlja ukupan broj sekundi koje su protekle od ponoći 1.januara 1970.godine, po griničkom vremenu, što je poznato i kao *Unix-ova epoha*.
- Izgleda čudno... ali ovo je standard ☺

# Prednosti i nedostaci Unix formata

## • Prednost:

- Čuvanje podataka o datumu i vremenu u sažetom obliku
- Na ovaj format ne utiče problem “milenijumske bube” (Y2K) koji ugrožava neke druge sažete ili skraćene formate za datum.

## • Nedostaci:

- Pomoću 32-bitne vrednosti je ograničen opseg datuma: softver ne može da evidentira događaje pre 1902.god. ili posle 2038.god. (Windows ne podržava negativan opseg!)

## • Funkcija date() i mnoge druge PHP-ove funkcije koriste ovaj standard, čak i kad se PHP koristi na serveru koji je pod Windows-om.

# Funkcija za konverziju u Unix format

- `int mktime ([int sati [, int minuti [, sekunde [, int mesec [, int dan [, int godina [, int letnje_r]]]]]])`
- Ova funkcija služi za pretvaranje podatka o datumu i vremenu u Unix-ovu vremensku oznaku.
- `letnje_r` je letnje računanje vremena i ima vrednost 1, ako se primenjuje, odnosno 0 ako se ne primenjuje, ili -1 ako ne zname.
- Zamerka:  
Redosled parametara je neintuitivan, ne dopušta izostavljanje parametara koji predstavlja vreme. Ako vreme nije bitno, možete zadati nule kao vrednosti parametara sati, minuti, sekunde.

# Funkcija getdate()



- Funkcija ima sledeći prototip:

```
array getdate ([int vremenskaoznaka])
```

- Funkcija razlaže opcionu vrednost vremenske oznake u niz, koja se sastoji od nekoliko komponenata, tj. podataka o datumu i vremenu.
- Ako funkciju pozovete bez datuma, dobićete vremensku oznaku tekućeg datuma i vremena.

# Komponente niza koji vraća getdate()

Ključ	Vrednost
seconds	Sekunde, numerička vrednost
minutes	Minuti, numerička vrednost
hours	Sati, numerička vrednost
mday	Dan u mesecu, numerička vrednost
wday	Redni broj dana u sedmici, numerička vrednost
mon	Mesec, numerička vrednost
year	Godina, numerička vrednost
yday	Redni broj dana u godini, numerička vrednost
weekday	Ime dana u sedmici, tekst
month	Puno ime meseca, tekst
0	Vremenska oznaka, numeričkog tipa

# Provera ispravnosti datuma

• Pomoću funkcije `checkdate()` možete utvrditi da li je datum ispravan (ovo je korisno kada korisnik unosi datum preko forme).

• Funkcija `checkdate()` ima sledeći prototip:

```
int checkdate (int mesec, int dan, int godina)
```

Primeri:    `checkdate(4,15,2012) //true`  
              `checkdate(4,31,2012) //false`

• Funkcija ispituje:

- Da li je godina ispravna celobrojna vrednost u opsegu od 0 do 32767?
- Da li je mesec celobrojna vrednost u opsegu od 1 do 12?
- Da li zadati dan postoji u zadatom mesecu?

# Konverzija datuma između PHP i MySQL

- U MySQL datum i vreme u formatu ISO8601
- ISO8601 datum počinje godinom: GGGG-MM-DD
- Na primer:  
10.april 2012. zadaje se kao 2012-04-10  
ili kao 12-04-10
- Zbog toga je za razmenu datuma potrebna konverzija PHP $\leftrightarrow$ MySQL
- Konverzija se vrši ili u PHP ili u MySQL

# DATE\_FORMAT() u MySQL-u

- Ako želite da konvertujete datume ili vreme u MySQL-u, koristite funkcije DATE\_FORMAT () i UNIX\_TIMESTAMP () .
- Funkcija DATE\_FORMAT () radi na sličan način kao ekvivalentna PHP-ova funkcija, ali date () koristi drugačije šablove za formate.
- Evropski format: DD-MM-GGGG  
ISO (MySQL) format: GGGG-MM-DD
- Prototip funkcije:  
`SELECT DATE_FORMAT(datumska_kolona,  
'%d. %m. %Y') FROM tabela;`

# Oznake u funkciji DATE\_FORMAT (1)

Kod	Opis
%M	Puno ime meseca
%W	Puno ime dana u nedelji
%D	Dan u mesecu, kao broj kome sledi tekstualni sufiks (Primer: 1st, 2nd, 3rd,...)
%Y	Godina kao četvorocifreni broj (Primer: 2012)
%y	Godina kao dvocifreni broj
%a	Skraćeno ime dana, do tri slova
%d	Dan u mesecu kao broj sa vodećom nulom
%e	Dan u mesecu kao broj bez vodeće nule
%m	Mesec kao broj sa vodećom nulom
%c	Mesec kao broj bez vodeće nule
%b	Skraćeno ime meseca sastavljenog od tri slova
%j	Redni broj dana u godini

# Oznake u funkciji DATE\_FORMAT (2)

Kod	Opis
%H	Čas u 24-časovnom formatu sa vodećom nulom
%k	Čas u 24-časovnom formatu bez vodeće nule
%h ili %l	Čas u 12-časovnom formatu sa vodećom nulom
%l	Čas u 12-časovnom formatu bez vodeće nule
%i	Minuti u času kao broj sa vodećom nulom
%r	Vreme u 12-časovnom formatu (Primer: hh:mm:ss [AM   PM] )
%T	Vreme u 24-časovnom formatu (Primer: hh:mm:ss)
%S ili %s	Sekunde u minutu kao broj sa vodećom nulom
%P	Deo dana, AM ili PM
%w	Redni broj dana u nedelji kao broj u opsegu od 0 (nedelja) do 6 (subota)

# Funkcija UNIX\_TIMESTAMP()



- Funkcija UNIX\_TIMESTAMP() radi na sličan način ali pretvara vrednosti iz kolone datumskog tipa u Unix-ovu vremensku oznaku.
- Prototip funkcije:

```
SELECT UNIX_TIMESTAMP(datumska_kolona)
FROM tabela;
```

- Vraća datum u vidu vremenske marke, pa se u PHP-u može koristiti po želji.
- Proračuni i poređenja datuma su jednostavniji kada su u Unix-ovom formatu!

# Primer - Broj godina



- Na osnovu unetog datuma rođenja u PHP kodu, izračunati starost osobe u godinama.

# Primer - Rešenje



```
<?php
$dan=18;
$mesec = 9;
$godina = 1988;

//formiramo datum rođenja u Unix formatu
$rodjendan = mktime(0,0,0,$mesec,$dan,$godina);

//formiramo tekuci datum u Unix formatu
$trenutno = time();

$godine_unix = $trenutno - $rodjendan; //razlika

//konverzija iz sekundi u godinu
$godine = floor($godine_unix / (365*24*60*60));
echo "Broj godina je $godine";

?>
```

# 5) Višekratna upotreba koda

- Iskazi require() i include()
- Definisanje sopstvenih funkcija
- Pozivanje po referenci i vrednosti
- Rekurzija

Jedan od ciljeva dobrih programera:

Koristiti što više napisan kod,  
umesto da pišemo novi kod.

# Upotreba require() i include()

- Pomoću iskaza require(), odnosno include() u PHP skript se može učitati datoteka. Ta datoteka može da sadrži iskaze PHP-a, tekst, HTML oznake, PHP-ove funkcije ili klase.

# Primer



Datoteka dodatak.php ima sledeći sadržaj:

```
<?php
 echo 'Ovo će biti ubaceno.
';
?>
```

Datoteka main.php ima sledeći sadržaj:

```
<?php
 echo 'Ovo je glavni fajl.
';
 require('dodatak.php');
 echo 'Ovde je kraj!
'
?>
```

# Šta je rezultat?



- Izraz `require()` umeće u skript sadržaj datoteke dodatak.php koju smo zadali.
- Kada pokrenemo skript, prvo se iskaz

```
require('dodatak.php');
```

zamenjuje sadržajem navedene datoteke, pa se tek onda skript izvrši.
- Razlika između require i include:  
Izraz `require()` je identičan kao `include()` osim što on daje grešku (`E_COMPILE_ERROR`), prilikom nemogućnosti da učita fajl, za razliku od `include` koji u slučaju ne učitavanja fajla prikazuje samo upozorenje (`E_WARNING`), tako da se skript nastavlja učitati dalje, preskačući taj skript koji se umeće.

# Primer šablonu za veb strane

The screenshot displays a website template with a blue header bar at the top. Below it is a red header section containing a logo (a white square with a blue 'T'), the company name 'Drazen Telekom' in white, and another logo (a blue square with a white 'T') on the right. The main content area has a white background with a blue navigation bar at the top. The navigation bar includes icons and text for 'Home', 'Servisi', 'Cenovnik', and 'Kontakt'. Below the navigation bar, a welcome message 'Dobrodosli! Najjeftinije telefoniranje u Srbiji!!!' is displayed. A blue footer bar at the bottom contains the text '© Telekom' and 'pogledajte nasu'.

Korisno je da ovu veb stranicu podelimo na delove:

- deo pre sadržaja - zaglavlje i meni (header.inc)
- dinamički deo sa sadržajem (home.php)
- deo iza sadržaja (footer.inc)

Datoteke header.inc i footer.inc sadrže kod koji ćemo upotrebljavati i na drugim stranicama.

# Iskazi require\_once i include\_once

```
//nekaNovaStrana.php
<?php
 require ('header.inc');
 require_once ('header.inc');
?>
```

- Iskaz `require_once(nazivSkripte)` je identičan kao `require`, ali kod nje PHP proverava da li je već skripta učitana, i ako je učitana ne treba je ponovo učitavati.
- Iskaz `include_once(nazivSkripte)` je identičan kao `include` ali slično kao i `require_once` proverava da li je skripta već jednom učitana u okviru skripte u kojoj se poziva

# Primer šablonu za veb strane

```
//home.php
<?php
 require('header.inc');
?>
<p>Dobrodosli! Najjeftinije telefoniranje u Srbiji!!!</p>
<?php
 require('footer.inc');
?>
```

- Savet je da nastavak imena datoteka čiji se sadržaj umeće u druge datoteke bude .inc (skraćeno od include). Takođe se preporučuje da se datoteke za umetanje postave izvan stabla veb dokumenata kako bi se sprečilo neovlašćeno čitanje vašeg izvornog koda.

# Funkcije



- Funkcija je samostalan modul koji propisuje način pozivanja funkcije, obavlja zadatak i eventualno vraća rezultat.
- Pozivanje funkcija      `moja_funkcija () ;`
- Pozivanje nedefinisanih funkcija => greška!
- Nema razlike između malih i velikih slova kod funkcija u PHP  
(Funk, FUNK, funk... je ista funkcija)

# Osnovna struktura funkcije



- Deklaracija počinje sa `function`, iza koje slede ime funkcije i parametri, a zatim i kod koji se izvršava kada je funkcija pozvana
- Imena funkcija:
  - Nova funkcija ne sme da ima isto ime kao neka postojeća
  - Ime funkcije može da bude sastavljeno od slova, cifara i donje crte `_`
  - Ime funkcije ne može počinjati cifrom
  - PHP ne dozvoljava preklapanje ugrađenih funkcija!!
- Može da postoji jedan ili više parametara, ali ne moraju svi biti obavezni.

# Primer - tabela



```
function napravi_tabelu($nesto)
{
 echo '<table border = 1>';
 reset($nesto); // Pokazivac na pocetak niza
 $vrednost = current($nesto);
 while ($vrednost) {
 echo "<tr><td>$vrednost</td></tr>\n";
 $vrednost = next($nesto);
 }
 echo '</table>';
}
$moj_niz = array('Prvo polje.', 'Drugo polje.', 'Treće
 polje.');
napravi_tabelu($moj_niz);
```

# Funkcije - tip povratne vrednosti

- PHP 7 uvodi i mogućnost definisanja povratne vrednosti funkcije, u vidu sledećih tipova:

- int
- float
- boolean
- string
- interfaces
- array
- callable

```
<?php
 declare(strict_types = 1);
 function dohvatiVrednost(int $broj): int {
 return $broj;
 }
 print(dohvatiVrednost(5));
?>
```

# Opseg važenja



- Opseg važenja promenljive određuje gde je promenljiva vidljiva i upotrebljiva.
- Promenljive deklarisane unutar funkcije čiji je opseg od mesta deklarisanja do kraja funkcije (lokalne promenljive - funkcijски opseg važenja).
- Promenljive deklarisane izvan funkcije čiji je opseg od mesta deklarisanja do kraja datoteke (globalne promenljive - globalni opseg važenja).
- Pomoću reči `global` može se ručno zadati globalni opseg važenja za promenljivu definisani uнутар funkcije.

# Prenos parametara po vrednosti

- Engl. *pass by value*
- Kada funkciji prosledite vrednost parametra, unutar funkcije se pravi nova promenljiva koja sadrži prosleđenu vrednost - kopija originala.
- Može slobodno da se menja,  
dok izvorna promenljiva ostaje nepromenjena.

# Prenos parametara po referenci

- Engl. *pass by reference*
- Kada se parametar prosledi funkciji, ona ne pravi novu promenljivu već preuzima referencu na originalnu promenljivu.
- Referenca se ponaša kao promenljiva, upućuje na original i nema sopstvenu vrednost (svaka izmena reference se odnosi i na original).

```
function inkrement(&$vr, $ink=1) {
 $vr = $vr + $ink;
}
```

```
$a = 10;
echo $a.'
';
inkrement($a);
echo $a. '
';
```

# Povratak iz funkcije



- Izvršenje funkcije se prekida sa `return`

```
function veci($x, $y) {
 if(!isset($x) || !isset($y)) {
 echo 'Prosledite dva broja';
 return;
 }
 if ($x>=$y)
 echo $x;
 else
 echo $y;
}
```

 **isset()** utvrđuje da li je promenljiva napravljena i da li ima vrednost

```
$a = 1;
$b = 2.5;
$c = 1.9;
veci($a, $b);
veci($c, $a);
veci($d,$a);
```

# Rekurzija



- Rekurzivne funkcije - one koje pozivaju samu sebe
- Prednosti: kraći kod, elegantnije rešenje
- Nedostaci: opterećenje memorije

```
function obrnuto_r($str)
{
 if (strlen($str)>0)
 obrnuto_r(substr($str, 1));
 echo substr($str, 0, 1);
 return;
}
```

```
obrnuto_r('Zdravo');
obrnuto_r('dravo');
obrnuto_r('ravo');
obrnuto_r('avo');
obrnuto_r('vo');
obrnuto_r('o');
obrnuto_r("");
```



# Pitanja?

Hvala!