Web programiranje

I GRUPA

1. Uvod u PHP (istorija, način izvršavanja, prednosti, primeri, instalacija)

- PHP je **server-side** programski jezik koji se koristi za pravljenje dinamičkih (sadržaj se može automatski promeniti svaki put kada se stranica pogleda), interaktivnih (reaguje na unos korisnika) veb stranica. Server-side znači da se PHP skripte ili programi uglavnom pokreću unutar web browser-a, PHP je interpretirani jezik – PHP skriptu procesuje PHP engine svaki put kada se skripta pokrene.

- PHP programi rade na veb serveru i serviraju veb stranice posetiocima na zahtev. Ugrađeni PHP kod unutar HTML veb stranica omogućava veoma lako i brzo kreiranje dinamičkog sadržaja. Druge WP tehnologije: ASP (Active Server Pages), ASP.NET, Perl, Java, Python, Ruby, Cold Fusion

Pokretanje PHP skripte: (način izvršavanja)

- Posetilac zatraži veb stranicu klikom na link ili upisivanjem URL-a stranice u browser address bar (posetilac tom prilikom može i da pošalje podatke veb serveru). Veb server prepoznaje da je zahtevani URL ustvari PHP skripta i daje uputstva PHP engine-u da obradi i pokrene skriptu. Kada se skripta izvrši, obično pošalje HTML stranicu veb serveru, koju korisnik vidi na svom ekranu.

Zadaci PHP-a:

- Čitanje i obrada sadržaja veb forme koju je poslao posetilac
- Čitanje, pisanje i kreiranje fajlova na veb serveru
- Rad sa podacima iz baze podataka koja je na veb serveru
- Hvatanje i obrada podataka sa drugih veb stranica
- Generisanje dinamičkih grafika kao što su grafici i obrađene slike

Primeri PHP skripti:

- Veb forumi koji omogućavaju posetiocima da postavljaju poruke i diskutuju o temama
- Search engines koji omogućavaju ljudima da pretražuju sadržaje veb sajtova ili baza podataka
- Sistemi za upravljanje sadržajem i blogovi koji omogućavaju webmasterima da kreiraju sajtove lako uz minimalno tehničko znanje
- Webmail aplikacije, omogućavaju ljudima da šalju i primaju e-poštu koristeći svoj web browser
- Online prodavnice, omogućavaju kupcima da kupuju proizvode i servise preko interneta

Prednosti PHP-a:

1. Popularnost:
    - veliki broj internet servis provajdera i web hosting kompanije podržavaju PHP
    - stotine hiljada programera koriste PHP
    - prijavljeno je da nekoliko miliona sajtova ima PHP
    - dokumentacija www.php.net
2. Cross-platform
    - OS: Windows, Linux, Mac OS X...
    - Web serveri: Apachem Internet Information Server, Zeus...
    - Mogućnost razvijanja i testiranja PHP veb sajtova na jednoj postavci (setup) a zatim je primeniti (deploy) na drugom tipu sistema

Instalacija:

- Softver potreban za PHP: Web server softver (npr. Apache ili IIS), PHP server module instaliran na istom računaru (on pokreće PHP skripte), opciono database server (npr. MySQL, SQL Server...), LAMP za Linux, WAMP za windows i MAMP za Mac OS

- Drugi načini za pokretanje PHP skripti obuhvataju pokretanje PHP-a sa drugim veb serverima (IIS, GWS, nginx...), kompajliranje PHP-a sa C kompajlerom, pokretanje PHP-a na daljinu (remote)

Istorija:

- Rasmus Lerdorf 1994. godine kreirao je **PHP** (Personal Home Page) i skup jednostavnih alata kodiranih u C jeziku
- **PHP verzija 2** objavljena za širu javnost 1995. godine
- **PHP verzija 3** objavljena u junu 1998. godine (Rasmus Lerdorf, Zeev Suraski i Andi Gutmans), većina koda je prepravljena
- **PHP 4**, objavljena u maju 2000. godine
    - PHP izvorni kod prepravljen opet
    - Zend Engine (**Ze**ev and A**nd**i)
    - Svojstvo upravljanja sesijama, baferovanje izlaza (output buffering), bogatiji izvorni jezik, širi izbor Web server platformi...
    - Još uvek relativno loša OOP implementacija
- **PHP 5**, objavljen u julu 2004
    - private i protected članovi klasa; final, private, protected i static metode; apstraktne klase; interfejsi; standardizovana konstruktor/destruktor sintaksa
- **PHP 6**, objavljene beleške
- **PHP 7**, objavljeno u decembru 2015. godine U razvoju

2. Osnovni elementi jezika (promenljive, tipovi, operatori, izrazi, konstante)

Promenljive (Variables):

- Omogućavaju skladištenje i izmenu podataka u skriptama. Promenljiva je kontejner koji sadrži određenu vrednost.

- Recimo za primer umesto echo 2+2 stavimo echo $x+$y, tako je skripta korisnija i fleksibilnija

- Imenovanje promenljivih ima svoja pravila: Sva imena promenljivih počinju znakom $, prvi karakter posle $ mora biti slovo ili znak _ dok ostatak karaktera može biti kombinacija slova, brojeva ili donjih crta bez ograničenja dužine

- Deklaracija: $my_var; inicijalizacija: $my_var = 3; Ako se ne inicijalizuje onda ima default vrednost null

Tipovi podataka (Data types):

- Koji tipovi su dostupni u PHP-u i kako testirati i promeniti tip

- Tip promenljive određuje koje operacije se mogu vršiti nad njom i koliko memorije će biti potrebno za nju, PHP podržava četiri skalarna tipa podataka i dva složena tipa

- **Skalaran tip** – sadrži samo jednu vrednost, tu spadaju **Integer**, **Float**, **String** i **Boolean**

- **Složen tip** – može sadržati više od jedne vrednosti, tu spadaju **nizovi** i **objekti**

- PHP takođe podržava i dva specijalna tipa podatka (nazivaju se specijalnim jer nemaju vrednost već samo značenje), tu spadaju **resursi** (referenca ka eksternom resursu npr bazi ili fajlu) i **null** (null kao vrednost, promenljiva nema nikakvu određenu vrednost)

- **PHP je loosely-typed jezik**, automatski konvertuje tip podatka promenljive u zavisnosti od konteksta u kome se ta promenljiva koristi, za proveru tipa promenljive koristi se funkcija *String* ***gettype($variable)****,* može se koristiti i:

Function	Description
is_int(*value*)	Returns true if *value* is an integer
is_float(*value*)	Returns true if *value* is a float
is_string(*value*)	Returns true if *value* is a string
is_bool(*value*)	Returns true if *value* is a Boolean
is_array(*value*)	Returns true if *value* is an array
is_object(*value*)	Returns true if *value* is an object
is_resource(*value*)	Returns true if *value* is a resource
is_null(*value*)	Returns true if *value* is null

- Tip promenljive se može promeniti korišćenjem funkcije ***settype($variable, "tip")*****,** pored ove funkcije može se koristiti i **kastovanje** npr. (string) $variable, razlika između ova dva je što kastovanje ne menja tu promenljivu a settype menja.

Function	Description
(int) *value* or (integer) value	Returns *value* cast to an integer
(float) *value*	Returns *value* cast to a float
(string) *value*	Returns *value* cast to a string
(bool) *value* or (boolean) value	Returns *value* cast to a Boolean
(array) *value*	Returns *value* cast to an array
(object) *value*	Returns *value* cast to an object

Function	Description
intval(*value*)	Returns *value* cast to an integer
floatval(*value*)	Returns *value* cast to a float
strval(*value*)	Returns *value* cast to a string

Operatori:

- Omogućavaju izmenu informacija. Operator je simbol koji menja jednu ili više vrednosti uglavnom proizvodeći novu vrednost u procesu. Vrednosti i promenljive koje se koriste sa operatorom se nazivaju *operandi*.

- Operatori u PHP-u se mogu grupisati u 10 tipova:

1. Aritmetički operator (obične aritmetičke operacije kao što su sabiranje i oduzimanje)
   - + sabiranje, - oduzimanje, * množenje, / deljenje, % modul
2. Operator dodele (dodeljuje vrednost promenljivoj)
   - =, +=, -=
3. Bitwise (izvodi operacije nad bitovima)
   - Za bit sa vrednošću 1 se kaže da je postavljen, a sa vrednošću 0 da nije postavljen (set i unset)
   - & (and), | (or), ^ (xor), ~ (not), << (pomeri ulevo), >> (pomeri udesno)
   - Bitwise operatori se obično koriste da se vrednosti spoje zajedno i naprave bit masku
4. Operator poređenja (poredi vrednosti u boolean stilu tj. true ili false se vraća)
   - == (jednako, ista vrednost), === (identično, isti tip i vrednost), != ili < > (nije jednako), !== (nije identično), < (manje), > (veće), <= (manje ili jednako), >= (veće ili jednako)
5. Error Control (utiče na error handling)
6. Operator izvršenja (vrši izvršavanje komandi kao da su shell komande)
7. Incrementing/decrementing (povećava ili smanjuje vrednost promenljive)
   - Dodaje ili oduzima vrednost 1, ++$x (doda 1 pa vrati rezultat), $x++ (vrati x pa doda 1), --$x, $x--
8. Logički (Boolean operatori poput **and, or,** i **not** koji se koriste da uključe ili isključe)
   - Rade na boolean vrednostima, **PHP smatra da su sledeće vrednosti false: doslovna vrednost false, integer 0, float 0.0, prazan string " ", string "0", niz sa 0 elemenata, null, SimpleXML objekat koji je kreiran od praznog XML taga, sve ostalo je true**

- && (and), and, || (or), or, xor, ! (not), razlika između && i and je u tome što && ima veći prioritet a and manji (isto važi i za || i or)

9. String (vrši konkatenaciju stringova – u PHP-u to ide preko znaka . , postoji samo jedan string operator)
10. Niz (Array, vrši operacije nad nizovima)

- **Prioriteti nekih operatora:**

Precedence of Some PHP Operators (Highest First)
++ - - (increment/decrement)
(int) (float) (string) (array) (object) (bool) (casting)
! (not)
* / % (arithmetic)
+ - . (arithmetic)
< < = > > = < > (comparison)
== != === !== (comparison)
& & (and)
|| (or)
= += - = *= /= .= %= (assignment)
and
xor
or

Konstante:

- Korisne za skladištenje podataka koji se **ne** menjaju u skripti. Imena konstanti ne počinju sa $ pored toga, što se tiče imenovanja, važi sve kao i za promenljive. Najbolje je koristiti sva velika slova za imenovanje konstanti. Za definisanje konstanti koristi se funkcija ***define()***

- Primer: define("MY_CONSTANT", 19); definisana je konstanta sa vrednošću 19

Izrazi:

- Izraz u PHP-u je bilo šta što procenjuje vrednost, to može biti kombinacija vrednosti, promenljivih, operatora i funkcija, primer: $x + $y je izraz

3. Naredbe grananja i petlje

- U **naredbe grananja** spadaju if, elseif, else i switch (koji se koriste isto kao i u javi, sa tim da je elseif spojeno, u switch-u posle default ide print "neki default tekst")

- Ternarni operator: (expression1) ? expression2 : expression3; ako je exp1 true izvršiće se exp2, u suprotnom exp3

- U **petlje** spadaju while, do while i for (koriste se isto kao i u javi), while se može koristiti i umesto for-a (kao i u javi). U PHP-u se inicijalizacija može izostaviti npr. **for (; $i < = 10; $i++) {// Loop body here}**, tu se očekuje da je promenljiva inicijalizovana pre petlje, a možemo izostaviti i sve elemente npr. **for (; ;) {// Infinite loop}** što predstavlja beskonačnu petlju i iz nje se može izaći samo koristeći break naredbu, ovi primeri pokazuju fleksibilnost for petlje u PHP-u tj. mogućnost izostavljanja pojedinih delova ukoliko su oni nepotrebni.

- **break** se koristi za izlazak iz petlje i vraća nas na prvu liniju koda van petlje iz koje smo izašli dok **continue** završava trenutnu iteraciju i prelazi na sledeću, primeri: (levo za break desno za continue)

```
$randomNumber = rand( 1, 1000 );
for ( $i=1; $i < = 1000; $i++ ) {
   if ( $i == $randomNumber ) {
        echo "Hooray! I guessed the random
        number. It was: $i < br / > ";
        break;
   }
}
```

```
for ( $i=1; $i < = 10; $i++ ) {
   if ( $i == 4 ) continue;
   echo "I ' ve counted to: $i < br / > ";
}
echo "All done!";
```

Važna napomena za break: break prihvata opcioni numerički argument koji govori iz koliko petlji treba da izađe, korisno kada imamo for u for-u (ugnježdene petlje) i želimo da izadjemo iz oba, možemo samo napisati break 2;

```
// Break out of the outer loop when $units == 5
for ( $tens = 0; $tens < 10; $tens++ ) {
    for ( $units = 0; $units < 10; $units++ ) {
      if ( $units == 5 ) break 2;
      echo $tens . $units . " < br / > ";
    }
}
```

Primer za break 2;

4. Stringovi

- String je niz karaktera, veb se zasniva na string podacima, HTML i XHTML stranice sastoje se od nizova običnog teksta.

- PHP ima skoro 100 različitih funkcija za manipulaciju stringovima. String funkcije mogu da se koriste za:

- Pretragu teksta unutar stringa
- Zamenu nekog teksta unutar string-a sa nekim drugim tekstom
- Formatiranje stringova tako da su lakši za čitanje ili rad
- Kodiranje i dekodiranje stringova koristeći popularne formate kodiranja

- Za stringove se mogu koristiti jednostruki (‘ ‘) i dvostruki navodnici (“ ”). Oni rade drugačije. Ako je string unutar jednostrukih navodnika PHP koristi taj string tačno onako kako je napisan (ako recimo koristimo tag <pre>) dok dvostruki navodnici imaju neke dodatne funkcionalnosti:

a) Sva imena promenljivih unutar stringa se parsiraju i zamenjuju vrednošću promenljive, primer:

```
$name = "Milan";                    //name je promenljiva sa vrednošću Milan
$message = "Zdravo, $name!";        //Kada napišemo "Zdravo, $name!", PHP zamenjuje $name sa "Milan"
echo $message;                      //Kada se string prikaže, rezultat je: Zdravo, Milan!
```

U ovom procesu:

Promenljiva unutar niza ($name) se analizira (parsing), tj. prepoznaje se kao promenljiva.
Zatim se zamenjuje svojom vrednošću (u ovom slučaju "Milan").

b) Možemo uključiti specijalne karaktere u string tako što ćemo ih izbeći znakom \, primer: $text = "\”tekst\””

Sequence	Meaning
\n	A line feed character (ASCII 10)
\r	A carriage return character (ASCII 13)
\t	A horizontal tab character (ASCII 9)
\v	A vertical tab character (ASCII 11)
\f	A form feed character (ASCII 12)
\\	A backslash (as opposed to the start of an escape sequence)
\$	A $ symbol (as opposed to the start of a variable name)
\"	A double quote (as opposed to the double quote marking the end of a string)

- Ako želimo da ispišemo vrednost promenljive direktno pre, posle ili između nekih karaktera, recimo da je promenljiva u pitanju $cat, ne možemo samo napisati $cats jer je to neka druga promenljiva, već moramo dodati zagrade ${cat}s ili {$cat}s

- String promenljivoj možemo dodeliti i vrednost bilo kog izraza:

$myString = ($x > 100) ? “Big number” : “Small number”;

- ***heredoc*** sintaksa je ekvivalenta korišćenja duplih navodnika, dok *newdoc* funkcioniše isto kao i jednostruki navodnici.

Heredoc: omogućava kreiranje višelinijskih stringova na čitljiviji način bez korišćenja navodnika i specijalnih karaktera

```
$myString = <<<DELIMITER          // DELIMITER je tekst koji se koristi za razdvajanje, tradicionalno
(insert string here)              // se koriste sva velika slova, sadrži samo slova, brojeve i _
DELIMITER;                        // mora počinjati sa slovom ili _
```

Primer:

```
$myString = <<<TEXT
Ovo je primer višelinijskog stringa
koji može uključivati specijalne karaktere kao što su "navodnici"
bez potrebe za izbegavanjem karaktera.
TEXT;
```

Newdoc: Sintaksa i primena su iste kao i kod heredoc jedina razlika je što se koriste jednostruki navodnici kod delimiter-

```
$myString = <<<’DELIMITER’
(insert string here)
DELIMITER;
```

- Funkcijom ***strlen()*** možemo odrediti dužinu stringa, prima promenljivu i vraća njen broj karaktera, dok str_word_count() broji reči u rečenici i uklanja interpunkcijske znakove prilikom brojanja.

- $character = $string[index]; se koristi za pristup karakteru na određenoj poziciji ili indeksu

- Funkcija ***substr()*** služi za izdvajanje niza karaktera iz stringa, ova funkcija uzima kao parametre string nad kojim će raditi, poziciju od koje počinje izvlačenje (ako tu stavimo negativan broj brojaće se od kraja stringa a ne od početka) i

broj karaktera koji se izvlače (ako se ovde iskoristi negativan broj preskače se toliko karaktera sa kraja stringa, ovaj parametar je opcioni – ako se izostavi izvlači se od početne pozicije do kraja stringa)

Primer:

```
$myString = “Hello, world!”;

echo substr($myString, 0, 5 );      // Displays ‘Hello’

echo substr($myString, -1 ) ;       // Displays ‘!’
```

- Za **<u>traženje nekog podstringa</u>** koriste se funkcije:

- **strstr()** – govori da li se neki tekst nalazi u stringu, uzima parametre string i tekst za pretragu, ako je tekst pronađen vraća deo stringa od početka pronađenog teksta do kraja stringa ili false ako tekst nije pronađen

  ```
  $myString = “Hello, world!”;
  echo strstr($myString, “wor”);              // Displays ‘world!’
  ```

  Postoji i opcioni boolean parametar, po default-u je on false ali ako se stavi true onda funkcija vraća deo od početka stringa do karaktera pre pronađenog teksta

  ```
  $myString = “Hello, world!”;      echo strstr( $myString, “wor”, true ); // Displays ‘Hello, ‘
  ```

- **strpos()** – vraća indeks pozicije *<u>prvog</u>* pojavljivanja teksta, uzima parametre string i tekst za pretragu, ako je tekst pronađen vraća indeks prvog karaktera teksta ako tekst nije pronađen vraća false

  ```
  $myString = “Hello, world!”;
  echo strpos( $myString, “wor” ); // Displays ‘7’
  echo strpos( $myString, “xyz” ); // Displays ‘’ (false)
  ```

  Kada proveravamo da li je nešto false moramo staviti === u slučaju da nam se vrati prazan string

  Može imati opcioni parametar, indeks početka pretrage

  ```
  $myString = “Hello, world!”;

  echo strpos( $myString, “o”, 5 ); // Displays ‘8’
  ```

- **strrpos()** – funkcioniše slično kao i strpos ali vraća indeks pozicije *<u>poslednjeg</u>* pojavljivanja teksta

  ```
  $myString = “Hello, world!”;

  echo strpos( $myString, “o” ) ;     // Displays ‘4’
  ```

  Postoje case-insensitive verzije ***stripos()*** i ***strripos()***

- **substr_count()** – govori koliko se puta dati tekst pojavljuje u stringu, parametri su string i tekst, vraća se int, može se ubaciti treći opcioni parametar indeks početne pozicije pretrage i cetvrti parametar koliko karaktera će funkcija pretražiti pre nego što odustane
- **strpbrk()** – pretražuje string tražeći bilo koji od karaktera iz zadate liste karaktera, vraća deo stringa od prvog karaktera koji se poklapa do kraja stringa, ako se ni jedan karakter ne pojavi u stringu, vraća se false

  ```
  $myString = “Hello, world!”;
  echo strpbrk( $myString, “abcdef” ); // Displays ‘ello, world!’
  ```

- **Menjanje teksta unutar stringova:** (case-sensitive su sve)

  - **str_replace() –** menja sve pojave teksta u stringu, uzima tri parametra: string koji se pretražuje, string zamena i string kroz koji se pretraga vrši, vraća kopiju originalnog stringa u kojoj su sve instance stinga koji se pretražuje izmenjene sa odgovarajućim stringom

    ```
    $myString = “That is a bird”;
    echo str_replace(“bird”, “plane”, $myString);
    // Displays “That is a plane”
    ```

    Može se proslediti i opcioni četvrti element varijabla u kojoj će stojati broj izmena

    ```
    echo str_replace(“bird”, “plane”, $myString, $num);

    echo “The text was replaced $num time”;

    // Displays “The text was replaced 1 time.”
    ```

    Postoji case-insensitive verzija ***str_ireplace()***

  - **substr_replace() –** menja specifičan deo stringa sa drugim stringom, prima tri parametra: string sa kojim radimo, tekst koji je zamena i indeks od koga počinjemo, vraća kopiju stringa sa zamenom bez menjanja originala

    ```
    $myString = “It was the best of times, it was the worst of times,”;
    echo substr_replace( $myString, “bananas”, 11 );
    // Displays “It was the bananas”
    ```

    Ako ne želimo da zamenimo sav tekst od zadatog početka do kraja stringa, možemo dodati opcioni četvrti parametar tj. broj koji govori koliko karaktera treba da se zameni

    ```
    $myString = “It was the best of times, it was the worst of times,”;

    echo substr_replace($myString, “bananas”, 19, 5 )”;

    // Displays “It was the best of bananas, it was the worst of times,”
    ```

    Ako je četvrti parametar negativan broj onda se broji od kraja niza

    ```
    $myString = “It was the best of times, it was the worst of times,”;

    echo substr_replace( $myString, “bananas”, 19, -20 );

    // Displays “It was the best of bananas the worst of times,”
    ```

    Ako je četvrti parametar nula onda se tekst dodaje na to mesto

    ```
    $myString = “It was the best of times, it was the worst of times,”;

    echo substr_replace( $myString, “really “, 3, 0 );

    // Displays “It really was the best of times, it was the worst of times,”
    ```

  - **strtr()** – menja specifičan karakter u stringu sa drugim karakterom, uzima tri argumenta: string sa kojim radimo, string koji sadrži listu karaktera koji treba da se zamene i string koji sadrži listu karaktera koji se stavljaju umesto njih, vraća izmenjenu kopiju stringa

    ```
    $myString = “Here’s a little string”;

    echo strtr( $myString, “ ‘”, “+-” );

    // Displays “Here-s+a+little+string”
    ```

Postoji case-insensitive verzija ***stristr()***

strtolower() – pretvara sva slova stringa u mala slova

strtoupper() – pretvara sva slova stringa u velika slova

ucfirst() – pretvara samo prvo slovo stringa u veliko slovo

lcfirst() – pretvara samo prvo slovo stringa u veliko slovo

ucwords() – pretvara prvo slovo svake reči u veliko slovo

- **Formatiranje stringova:**

printf() uzima string argument *format string* i obično jedan ili više dodatnih argumenata koji sadrži jedan ili više stringova koji se formatiraju pa štampa rezultat

```
printf( “Pi rounded to a whole number is: %d”, M_PI );

// Displays “Pi rounded to a whole number is: 3”

printf( “%d times %d is %d.”, 2, 3, 2*3 );

// Displays “2 times 3 is 6.”
```

Type Specifier	Meaning
b	Treat the argument as an integer and format it as a binary number.
c	Treat the argument as an integer and format it as a character with that ASCII value.
d	Treat the argument as an integer and format it as a signed decimal number.
e	Format the argument in scientific notation (for example, 3.45e+2).
f	Format the argument as a floating - point number, taking into account the current locale settings (for example, many European locales use a comma for the decimal point, rather than a period).
F	Format the argument as a floating - point number, ignoring the locale settings.

o	Treat the argument as an integer and format it as an octal number.
s	Format the argument as a string.
u	Treat the argument as an integer and format it as an unsigned decimal number.
x	Treat the argument as an integer and format it as a lowercase hexadecimal number.
X	Treat the argument as an integer and format it as an uppercase hexadecimal number.
%	Display a literal percent (%) symbol. This doesn’t require an argument.

- printf(“**%+d**”, 123); // Displays “+123” Minus se automatski stavlja, plus moramo ručno

- Padding tj. razmak dodajemo na sledeći način (primer): printf(“**%06d**”, 4567); // Displays “004567”, kad stavimo 0 broj se popunjava nulama da bi (u ovom primeru) došao do 6 cifara

Takođe se može koristiti i space

tj. razmak (primer):

```
print " < pre > ";
printf( "% 15s\n", "Hi" );
printf( "% 15s\n", "Hello" );
printf( "% 15s\n", "Hello, world!" );
print " < /pre > ";
```

```
Here ' s the result:
                    Hi
                 Hello
          Hello, world!
```

- Ne moramo koristiti samo 0 i razmake, može se dodati i proizvoljan padding karakter tako što stavimo apostrof (‘) i nakon njega karakter koji želimo da koristimo, primer: printf(“%’#8s”, “Hi”); // Displays “######Hi”

- Po default se popunjava sa leve strane, možemo namestiti da se popunjava sa desne tako što dodamo “–“ ispred broja

str_pad(), prima parametre string i širina konačnog stringa, funkcija vraća tekst koji je dopunjen sa desne strane koristeći prazna mesta tj. razmake, može se koristiti i opcioni treći argument koji predstavlja znak kojim će se string dopuniti

echo str_pad(“Hello, world!”, 20, “*”);

// Displays “Hello, world!*******”

echo str_pad(“Hello, world!”, 20, “123”);

// Displays “Hello, world!1231231”

Može se dodati i opcioni četvrti element koji se se sastoji od jedne od konstanti:

STR_PAD_RIGHT – za dopunjavanje stringa sa desne strane, to je po default, left-aligning

STR_PAD_LEFT – za dopunjavanje stringa sa leve strane, right-aligning

STR_PAD_BOTH – za dopunjavanje stringa i sa leve i sa desne strane, tekst je što više na sredini

Primer: echo str_pad(“Hello, world!”, 20, “*”, STR_PAD_BOTH);

// Displays “***Hello, world!****”

- Možemo navesti na koliko decimala se broj zaokružuje tako što dodamo “.” Ispred broja:

printf(“%.2f”, 123.4567); // Displays “123.46”, zaokruženo na dve decimale

Možemo ovo kombinovati sa padding-om: printf(“%012.2f”, 123.4567); // Displays “000000123.46”

Ako imamo stringove prikaže samo toliko koliko smo naveli: printf(“%.8s”, “Hello, world!”); // Displays “Hello, w”

- Važno je da argumenti koji se prosleđuju printf funkciji budu u tačnom redosledu, ako želimo da ručno promenimo redosled možemo to uraditi dodavanjem pozicije argumenta nakon svakog znaka % a pre znaka $ u nekom txt fajlu, to se naziva *argument swapping,* primer:

    Your %2$s contains %3$d unread messages, and %1$d messages in total.

sprintf() – ponaša se isto kao i printf(), jedina razlika je u tome što sprintf vraća string a prinf samo ispisuje rezultat

    $messageCount = sprintf(file_get_contents("template.txt"), $totalMessages, $mailbox, $unreadMessages);

"Trimovanje" tj. seckanje praznih mesta (white space) u stringu:

- **trim() –** briše prazna mesta sa početka i kraja stringa
- **ltrim() –** briše prazna mesta samo sa početka stringa
- **rtrim() –** briše prazna mesta samo sa kraja stringa

Kao argument prima string, mada možemo dodati i opcioni drugi argument: string karaktera koji se tretiraju kao white space (default white space karakteri su: "" (space), "\t" (tab), "\n" (newline), "\r" (carriage return), "\0" (a null byte), i "\v" (vertical tab))

Može se koristiti ".." da se specificira opseg karaktera

    $milton1 = "**1:** The mind is its own place, and in it self\n";

    echo ltrim($milton1, "**0..9:** "); //uklanja brojeve od 0 do 9, znak : i razmak, i to sa početka stringa

    The mind is its own place, and in it self //ovo je prikazano

-wordwrap() – uzima single-line string teksta i deli ga na više linija koristeći novi red ("\n"), nijedna linija ne prelazi 75 karaktera, mada može se promeniti dodavanjem drugog argumenta, dakle za primer wordwrap ($myString, 40); funkcija će uzeti string i na svakih 40 karaktera staviti novi red, kao treći argument može se staviti proizvoljni karakter koji će se koristiti umesto default novog reda

Postoji i četvrti opcioni karakter koji može biti true ili false. Ako je true onda funkcija uvek razdvoji baš na toj dužini što može prouzrokovati deljenje reči, zato je po default na false

-number_format() – lakši način za formatiranje brojeva, razdvajanje po hiljadama i zaokruživanje na određeni broj decimalnih mesta (drugi, opcioni argument kaže na koliko tačno decimala)

    echo number_format(1234567.89, 1); // Displays "1,234,567.9"

Dodavanjem još dva opciona argumenta možemo promeniti karaktere:

    echo number_format(1234567.89, 2, ",", " "); // Displays "1 234 567,89"

    (gde je , zamenjen sa praznim stringom, dakle zarez se u ovom slučaju briše)

5. Nizovi

- Niz je jedna varijabla koja može sadržati više vrednosti odjednom, vrednosti unutar niza se nazivaju elementi a svaki element je referenciran sopstvenim indeksom koji je jedinstven za taj niz, indeks se koristi za pristupanje elementu radi dalje obrade.

- Dve specifične karakteristike nizova čine ih dobrim za skladištenje puno podataka:

1. Niz može biti bilo koje dužine, dužina niza se može lako promeniti dodavanjem ili uklanjanjem vrednosti bilo kad
2. Lako je upravljati svim članovima niza odjednom (npr. prolazak kroz sve vrednosti manjajući ih usput, sortiranje nizova, pronalaženje vrednosti u nizu, spajanje dva niza itd.)

- PHP podrzava **dve vrste niza:**

1. **Indeksirani nizovi**: Nizovi u kojima je svaki element referenciran numeričkim indeksom, uglavnom počevši od 0
2. **Asocijativni nizovi**: Hash ili mapa, svaki element je referenciran string indeksom npr. kreiramo element niza: godina mušterije, i damo mu indeks “godina”

Kreiranje nizova:

-Najlakši način za kreiranje nizova je koristeći built-in kontrukciju **array()**

Primer za indeksirane nizove: $authors = array(“Steinbeck”, “Kafka”, “Tolkien”, “Dickens”);

Primer za asocijativne nizove: $myBook = array(“title” = > “The Grapes of Wrath”, “author” = > “John Steinbeck”, “pubYear” = > 1939);

- Da bi pristupili elementu indeksiranog niza pišemo ime promenljive sa indeksom elementa u kockastim zagradama:

```
$authors = array( “Steinbeck”, “Kafka”, “Tolkien”, “Dickens” );

$myAuthor = $authors[0]; // $myAuthor contains “Steinbeck”
```

- Da bi pristupili elementu asocijativnog niza koristimo string indekse:

```
$myBook = array( “title” = > “The Grapes of Wrath”, “author” = > “John Steinbeck”, “pubYear” = > 1939 );

$myTitle = $myBook[“title”]; // $myTitle contains “The Grapes of Wrath”
```

- Menjanje i kreiranje novog člana niza su slični javinom pristupu:

```
“ Tolkien ” to “ Melville “ :

$authors[2] = “Melville”;          //menjamo tolkien u melville

$authors = array( “Steinbeck”, “Kafka”, “Tolkien”, “Dickens” );

$authors[4] = “Orwell”;            //dodajemo novi član sa specifičnim indeksom

// U slučaju da želimo da samo dodamo element sa sledećim slobodnim indeksom u nizu, pišemo:

    $authors[] = “Orwell”;
```

- Funkcija **print_r()** štampa sve elemente niza, obično se koristi samo u svrhe debagovanja, prosleđuje se samo niz za štampanje: print_r($array); Štampa se u formatu **ključ = vrednost,** moguće je dodati i drugi parametar true da bi se vrednost čuvala u nekoj promenljivoj umesto samo štampala

```
$arrayStructure = print_r( $array, true );
```

```
echo $arrayStructure; // Displays the contents of $array
```

array_slice() izdvaja opseg elemenata iz niza, prosleđuje joj se niz, nakon njega pozicija prvog elementa u opsegu (počevši od 0), zatim broj elemenata koji se izdvajaju (ako ostavimo prazno ide do kraja niza), vraća novi niz koji sadrži kopije izdvojenih elemenata

```
$authors = array( “Steinbeck”, “Kafka”, “Tolkien”, “Dickens” );

$authorsSlice = array_slice( $authors, 1, 2 );

print_r( $authorsSlice );          // Displays “Array ( [0] = > Kafka [1] = > Tolkien )”
```

Ovo se može koristiti i u radu sa asocijativnim nizovima jer PHP pamti redosled elemenata u asocijativnom nizu.

Možemo dodati i četvrti opcioni argument, true, koji čuva originalne indekse tih elemenata (ne stavlja nove)

count() – prebrojava elemente u nizu i vraća broj elemenata, count() – 1 za poslednji element niza

- Možemo se kretati kroz elemente niza pomoću pokazivača jer PHP pamti redosled dodavanja elemenata u niz i ima pokazivač na prvi element niza

Function	Description
current()	Returns the value of the current element pointed to by the pointer, without changing the pointer position.
key()	Returns the index of the current element pointed to by the pointer, without changing the pointer position.
next()	Moves the pointer forward to the next element, and returns that element’s value.
prev()	Moves the pointer backward to the previous element, and returns that element’s value.
end()	Moves the pointer to the last element in the array, and returns that element’s value.
reset()	Moves the pointer to the first element in the array, and returns that element’s value.

- Sve ove funkcije vraćaju vrednost ili indeks elementa i false u slučaju da se element ne može pronaći. Ovde dolazimo do problema: **Šta ako je vrednost elementa u nizu false? U tom slučaju funkcija vraća false i nećemo znati da li smo dobili vrednost elementa ili je postojao problem pri dobavljanju vrednosti.**

- Funkcija **each()** vraća trenutni element niza i prebacuje pokazivač na sledeći element, vraća niz od četiti elementa u koje spadaju i ključ i vrednost elementa:

Element Index	Element Value
0	The current element’s key
“key”	The current element’s key
1	The current element’s value
“value”	The current element’s value

```
$myBook = array( "title" = > "The Grapes of Wrath",
                 "author" = > "John Steinbeck",
                 "pubYear" = > 1939 );
$element = each( $myBook );
echo "Key: " . $element[0] . " < br/ > ";
echo "Value: " . $element[1] . " < br/ > ";
echo "Key: " . $element["key"] . " < br/ > ";
echo "Value: " . $element["value"] . " < br/ > ";
```

This code displays:

```
Key: title
Value: The Grapes of Wrath
Key: title
Value: The Grapes of Wrath
```

- Primer kako se koristi each() da se pribavi element koji ima vrednost false:

```
$myArray = array( false );

$element = each( $myArray );

$key = $element["key"]; // $key now equals 0

$val = $element["value"]; // $val now equals false
```

- Za prolazak kroz nizove efikasno je koristiti **foreach** koji radi samo za nizove i objekte. Može se koristiti na dva načina: da dobavi samo vrednost svakog elementa ili vrednost i ključ elementa

```
foreach ( $array as $value ) {
        // (do something with $value here)}
// (rest of script here)
```

Pri svakoj iteraciji promenljiva $value se postavlja na vrednost trenutnog elementa

- Da bi smo dobavili i vrednost i ključ koristi se sledeća sintaksa:

```
foreach ( $array as $key = > $value ) {
        // (do something with $key and/or $value here}
// (rest of script here)
```

- Ako promenimo vrednosti u foreach zapravo ne menjamo vrednosti u originalnom nizu, ako baš želimo da promenimo vrednosti treba da koristimo reference umesto kopija što se postiže korišćenjem simbola &

```
foreach ( $array as & $value) { ... }
```

Dobra praksa je posle ciklusa uraditi **unset($value)** da bi se osigurali da je promenljiva $value izbrisana kada se ciklus završi i da više ne pokazuje ni na šta, kad bi kasnije promenili $value promenljivu u kodu izmenili bi poslednji element niza jer $value još uvek pokazuje na taj element, zato je dobro raditi unset da se taj potencijalni problem ne bi desio

Višedimenzionalni nizovi:

- Predstavljaju nizove unutar nizova tj. kada je član nekon niza ustvari drugi niz. Da bi smo pristupili određenom elementu nekog podniza radimo to sa dve [][], slično kao i u javi, npr. $myBooks[1]["title"] gde je myBooks neki niz, 1 je indeks elementa (dakle drugi podniz jer krećemo od 0 brojanje) i title predstavlja korišćenje ključa asocijativnog niza za pristup elementu tog podniza na indeksu 1

- Prolazak kroz višedimenzionalne nizove se vrši sa ugnježdenim ciklusima

Sortiranje nizova

- U PHP-u postoji 12 funkcija za sortiranje nizova: (svaka od njih za parametar prima samo niz i vraća true/false)

- **sort() i rsort()** – za indeksirane nizove, sort() sortira vrednosti niza u **rastućem redosledu** dok rsort() sortira u **opadajućem,** sort() i rsort() ***reindeksiraju niz*** što znači da dodaju nove indekse elementima dok ih sortiraju što znači da ako sortiramo asocijativni niz i koristimo ove funkcije on će postati indeksiran
- **asort() i arsort() –** za asocijativne nizove, rade isto kao i sort() i rsort(), razlika je u tome što čuvaju vezu između indeksa i vrednosti
- **ksort() i krsort() –** za sortiranje asocijativnih nizova po ključu a ne po vrednosti, pored toga funkcionišu slično kao i asort() i arsort()
- **array_multisort()** – može da sortira više nizova odjednom očuvavajući odnose između njih, kao parametre prima jedan ili više nizova za sortiranje, npr. array_multisort($titles, $authors, $pubYears); prvo sortira po naslovima, može se koristiti i za sortiranje multidimenzionalnih nizova gde se onda prosleđuje samo taj jedan niz, funkcija onda sortira taj niz po prvom podnizu

Dodavanje i uklanjanje elemenata

- PHP ima 5 funkcija za dodavanje i uklanjanje elemenata niza:

- **array_unshift() –** dodaje jedan ili više novih elemenata na početak niza, kao parametar prima niz u koji ubacujemo element i elemente koji želimo da ubacimo odvojene zarezima
- **array_shift() –** uklanja prvi element sa početka niza i vraća njegovu vrednost, kao parametar prima niz
- **array_push() –** dodaje jedan ili više novih elemenata na kraj niza, kao parametar prima niz i elemente odvojene zarezima, vraća dužinu novog niza
- **array_pop() –** uklanja poslednji element sa kraja niza i vraća njegovu vrednost, kao parametar šrima niz
- **array_splice() –** uklanja jedan ili više elemenata iz niza i/ili dodaje jedan ili više elemenata na bilo koje mesto u nizu tj. dozvoljava nam da uklonimo broj elemenata iz nekog opsega u nizu i zamenimo ga sa elementima iz drugog niza (i uklanjanje i zamena su opcioni), kao parametre uzima niz i poziciju prvog elementa (brojanje počinje od 0), potom možemo dodati opcioni parametar koji specificira koliko elemenata se uklanja (ako on nije zadat uklanja se sve od zadatog početka do kraja niza), može se dodati još jedan parametar, niz elemenata za ubacivanje, vraća niz izvučenih elemenata

array_merge() – spaja dva ili više niza zajedno u jedan veliki niz koji posle vraća, čuva vezu između ključeva asocijativnih nizova pa se zato može koristiti za dodavanje novog ključ-vrednost para u neki asocijativni niz, npr:

$myBook = array(“title” = > “The Grapes of Wrath”, “author” = > “John Steinbeck”, “pubYear” = > 1939);

$myBook = array_merge($myBook, array(“numPages” = > 464));

Ako na tom mestu već nešto postoji biće zamenjeno (overwritten), zato je korisna za ažuriranje asocijativnih nizova, kod indeksiranih nizova se ne zamenjuje na taj način nego se doda na kraj i doda mu se novi indeks. Ako funkciji samo prosledimo neki indeksirani niz ona će izvršini reindeksiranje

explode() – pretvara string u niz tako što uzme niz i podeli ga na delove u zavisnosti od delimiter-a i vrati niz

```
$fruitString=“apple,pear,banana,strawberry,peach”;

$fruitArray = explode( “,”, $fruitString );
```

- Možemo dodati i treći (opcioni) parametar koji ograničava koliko elemenata će biti u vraćenom stringu, u tom slučaju poslednji element sadrži sve što nije stalo (ceo ostatak stringa bude spojen u njega). Takođe možemo proslediti negativan broj tu, on predstavlja broj elemenata koji će se skroz odstraniti sa kraja stringa u nizu

implode() – spaja elemente nekog niza u jedan veliki string, npr. $fruitString = implode(“,”, $fruitArray);

list() – izvlači vrednosti niza u pojedinačne promenljive, funkcioniše samo sa indeksiranim nizovima, kombinuje se uglavnom sa each()

```
$myBook = array( “The Grapes of Wrath”, “John Steinbeck”, 1939 );

list( $title, $author, $pubYear ) = $myBook;
```

Primer sa each:

```
$myBook = array( “title” = > “The Grapes of Wrath”, “author” = > “John Steinbeck”, “pubYear” = > 1939 );

while ( list( $key, $value ) = each( $myBook ) ) {

        echo “ < dt > $key < /dt > ”;

        echo “ < dd > $value < /dd > ”; }
```

6. Funkcije

- Funkcija, koje se u nekim drugim jezicima naziva i subprogram, je samostalni blok koda koji obavlja određeni zadatak

- Funkcije su korisne da bi se izbeglo dupliranje koda, da bi se lakše otklonile greške u kodu, funkcije se mogu ponovo koristiti u drugim skriptama i mogu pomoći pri razbijanju velikog projekta

- Možemo menjati ime funkcije tako što njeno ime stavimo u neku promenljivu:

```
$squareRoot = “sqrt”;

echo “The square root of 9 is: “ . $squareRoot( 9 ) ;

echo “All done!“;
```

- Prilikom pravljenja sopstvenih funkcija možemo navesti parametre (slično kao i u javi) ali ovde se mogu navesti i *opcioni parametri* koji se definišu tako što se pored imena parametra stavi znak = i neka default vrednost koja će se proslediti ukoliko se taj parametar ne prosledi pri pozivu funkcije

```
function myFunc( $parameterName=defaultValue ) {

        // (do stuff here)}
```

- U PHP-u se pre definisanja funkcije ne navodi šta ona vraća nego samo postoji return unutar funkcije

```
function myFunc() {     // (do stuff here)

                        return value;   }    //gde value može biti bilo šta, return se može koristiti i bez vrednosti
```

- Promenljive se mogu kreirati unutar funkcija i njima ne možemo pristupiti van te funkcije, to su onda **lokalne promenljive**

```
function describeMyDog() {
    $color = "brown";
    echo "My dog is $color < br/ > ";
}

// Define my cat's color
$color = "black";

// Display info about my dog and cat
describeMyDog();
echo "My cat is $color < br/ > ";
```

result:

```
My dog is brown
My cat is black
```

Da lokalne promenljive ne postoje onda bi se, pri pozivu funkcije describeMyDog(), prepravila (overwrite) vrednost promenljive $color tj. vrednost "black" bi postala "brown" i dobili bi smo rezultat

My dog is brown

My cat is brown

- **Globalna promenljiva** je promenljiva kojoj možemo pristupiti bilo gde u skripti nebitno da li je unutar ili van funkcije. Promenljive koje su kreirane van funkcije su globalne u smislu da im možemo pristupiti bilo gde u kodu SEM unutar funkcije, ako želimo takvu promenljivu da koristimo unutar funkcije moramo pre njenog imena dodati ključnu reč ***global***

```
$myGlobal = "Hello there!";

function hello() {

        global $myGlobal;        //morali smo koristiti reč global da bi pristupili promenljivi $myGlobal

        echo "$myGlobal";

}

hello();  // Displays "Hello there!"
```

Takođe možemo deklarisati više od jedne globalne promenljive (kao što možemo raditi i za svaki drugi tip)

```
function myFunction() {

        global $oneGlobal, $anotherGlobal;

}
```

Još jedan način na koji možemo pristupiti globalnim promenljivama je koristeći niz **$GLOBALS**, ovaj niz predstavlja specijalan tip promenljive koji se naziva ***superglobal***, što znači da joj možemo pristupiti bilo gde u kodu BEZ korišćenja ključne reči *global*. Sadrži listu svih globalnih promenljivih gde su imena promenljivih smeštena u njene ključeve a vrednosti promenljivih smeštene u njene vrednosti

```
$myGlobal = "Hello there!";

function hello() {

        echo $GLOBALS["myGlobal"] ;    }

hello();    // Displays "Hello there!"
```

- Statične promenljive su još uvek lokalne funkciji u smislu da im se može pristupiti samo unutar funkcije. One pamte svoje vrednosti od jednog poziva funkcije do sledećeg, da bi smo deklarisali nešto kao statičko samo dodamo ključnu reč *static* i dodamo inicijalnu vrednost toj promenljivoj

Anonimne funkcije

- PHP dozvoljava kreiranje **anonimnih funkcija**, tj. funkcija koje nemaju ime, uglavnom se prave iz dva razloga:

- da bi se dinamički kreirale funkcije, možemo personalizovati kod unutar funkcije u trenutku kada je kreiramo
- da bi se kreirale kratkoročne funkcije za jednokratnu upotrebu (ovo se obično radi kada se radi sa built-in funkcijama koje zahtevaju da se kreira callback ili event handler funkcija sa kojom se radi)

- Za kreiranje anonimne funkcije koristi se ***create_function()*** koja očekuje dva argumenta: listu parametara odvojenih zarezima (ako ih ima) i kod za telo funkcije, prost primer:

```
$mode = "+";
$processNumbers = create_function( '$a, $b', "return \$a $mode \$b;" );    // znak \ je za izbegavanje karaktera
echo $processNumbers( 2, 3 ); // Displays "5"                               //to ne moramo raditi ako koristimo
                                                                            //jednostruke navodnike
```

- Logika sortiranja: skripta koristi PHP-ovu ***usort()*** funkciju da sortira niz reči, usort() očekuje niz za sortoranje i funkciju poređenja povratnog poziva (callback comparison function), sve funkcije poređenja moraju prihvatiti dve vrednosti $a i $b i vratiti jednu od sledeće tri vrednosti:

- Negativan broj ako je $a "manje od" $b
- Nula ako je $a "jednako sa" $b
- Pozitivan broj ako je $a "veće od" $b

To bi izgledalo ovako: usort($words, create_function('$a, $b', 'return strlen($a) - strlen($b);'));

Ovo se može uraditi i bez anonimne funkcije na sledeći način:

```
function sortByLength( $a, $b ) {
        return strlen( $a ) - strlen( $b );}
usort( $words, "sortByLength" );
```

- Kada izmenimo promenljive unutar funkcije radimo sa kopijama originalnih promenljivih i samim tim ih ne menjamo. Ako želimo da menjamo vrednosti moramo koristiti **reference**. One se prave tako što ispred imena dodamo &.

```
$myVar = 123;
$myRef = & $myVar;
$myRef++;
echo $myRef; // Displays "124"
echo $myVar; // Displays "124"
```

- Funkcije takođe mogu vraćati reference umesto vrednosti, to se postiže tako što se znak & stavi pre imena funkcije u definiciji.

```
function & myFunc(){
        // (do stuff)
        return $var; // Returns a reference to $var}
```

Još jedan primer:

```
$myNumber = 5;

function & getMyNumber() {

        global $myNumber;

        return $myNumber;}

$numberRef = & getMyNumber();

//gore je stavljen & jer želimo da $numberRef promenljiva takođe bude referenca, da smo ga izostavili
$numberRef bi dobio kopiju vrednosti vraćene od strane funkcije i onda promene na $numberRef neće uticati na
$myNumber jer nisu povezane referencom, pa bi rezultat bio 5 i 6 a ne 6 i 6

$numberRef++;

echo "\$myNumber = $myNumber"; // Displays "6"

echo "\$numberRef = $numberRef"; // Displays "6"
```

Rekurzivne funkcije

- Rekurzija se javlja kada funkcija poziva samu sebe, takav proces bi se beskonačno ponavljao ako se ne zaustavi, zato rekurzija mora imati neki uslov pod kojim bi se završila, taj uslov se još naziva i *base case* a deo funkcije koji se ponavlja je *recursive case*.

- Način na koji funkcioniše rekurzivna funkcija:

- Pozivni kod poziva rekurzivnu funkiju
- Ako je base case tj. krajnji uslov ispunjen, funkcija obavlja bilo koju potrebnu obradu i zatim prekida rad
- Inače, funkcija obavlja ono što je potrebno i onda poziva sebe da nastavi rekurziju

7. Klase i objekti

- Objekti se prosleđuju po referenci funkcijama.

Vidljivost atributa

Public – može da mu se pristupi iz bilo kog koda, bilo da je taj kod unutar ili van klase.

Private – može da im se pristupi samo iz koda unutar klase.

Protected – slične su private, ali svaka klasa koja nasleđuje određenu klasu može da im pristupi.

Bezbednije je praviti private svojstva i kreirati getter i setter metode za njihovo dobavljanje.

Deklaracija atributa(svojstava properties)

```
class MyClass {

        public $property1; // This is a public property

        private $property2; // This is a private property

        protected $property3; // This is a protected property }

        static public $staticko = 111;
```

class MyClass { public $widgetsSold = 123; }

Pristupanje svojstvu – atributu

$object->property;

Statička svojstva(properties)

- Statički članovi klase su nezavisni od bilo kog objekta koji se dobije od te klase.

- Da bi se pristupilo statičkom svojstvu koristimo ime klase potom operator :: pa onda naziv svojstva tj atributa

npr **MyClass::$staticko++;**

Konstante klase

- Konstante unutar klase se deklarišu sa ključnom rečju **const**

npr.

class MyClass { const MYCONST = 123; }

-Pristupanje konstanti klase ide putem imena klase i operatora :: kao i kod statičkih članova

echo MyClass::MYCONST;

Metode

Korišćenjem ključne reči function.

```
class MyClass {
    public function aMethod() {
    // (do stuff here)
    } }
```

-Tri nivoa vidljivosti: public, private and protected, ukoliko se izostavi podrazumevano je public.

Pozivanje metoda

Metod se poziva nad objektom te klase

```
class MyClass {
    public function hello() {
        echo “Hello, World!”; } }
$obj = new MyClass;
$obj->hello(); // Displays “Hello, World!”
```

Parametri i povratne vrednosti

```
public function aMethod( $param1, $param2 ) {
// (do stuff here)
    return true; }
```

Pristupanje svojstvima objekta iz metode

$this – koristi za poziv metode objekta iz druge metode istog objekta.

```
class MyClass {
        public $greeting = “Hello, World!”;
         public function hello() {
                echo $this->greeting; } }
$obj = new MyClass;
$obj-> hello(); // Displays “Hello, World!”
```

Statički metodi

```
class MyClass {
        public static function staticMethod() { // (do stuff here) } }
 MyClass::staticMethod();
```

Pristup statičkim metodima, atributima i konstantama unutar iste klase

- Da bi se pristupilo navedenom unutar metode iste klase može da se koristi ista sintaksa kao i izvan klase znači naziv klase operator :: i ime atributa

MyClass::MYCONST

-Druga opcija je korišćenje ključne reči ***self npr. self::MYCONST***

Type hinting

public function paint(**Car** $car, $color)

- Daje razumniji eror

- Radi i sa redovnim funkcijama, ne samo metodama

function showAll(**array** $items) {

- Nažalost Php samo podržava type hinting samo za objekte i nizove.

Koristiti funkcije kao što su is_string() , is_int()

Overloading sa __get(), __set(), and __call()

- Presreće pokušaje da pročita illi napiše svojstva(properties) objekta, ili da pozove njegove metode.

Primeri:

- Pozivajući kod čita vrednost $myObject → property, što zapravo uzrokuje da $myObject preuzme vrednost iz niza.
- Pozivajući kod postavlja $myObject -> anotherProperty na novu vrednost, ali u pozadini $myObject zapravo piše vrednost u polje u bazi

- Pozivajući kod poziva $myObject -> aMethod(). Ovaj meto zapravo ne postoji u $myObject, ali on presreće poziv i poziva drugu metodu

Tri magična metoda:

- __get() je pozvana kad god pozivajući kod pokuša da čita nevidljivo svojstvo objekta.
- __set() je pozvana kad god pozivajući kod pokuša da piše unutar nevidljivog svojstva objekta
- __call() je pozvana kad god pozivajući kod pokuša da pozove nevidljivi metod objekta

Nevidljivo može da znači: da svojstvo ili metod ne postoje u klasi ili su private ili protected

__get()

```
class Car {
  public function __get( $propertyName ) {
    echo "The value of '$propertyName' was requested < br / > ";
    return "blue";
  }
}
$car = new Car;
$x = $car-> color;
  // Displays "The value of 'color' was requested"
echo "The car's color is $x < br / > ";
  // Displays "The car's color is blue"
```

__set() - zahteva dva parametra: naziv svojstva i vrednost koja mu se postavlja, ne zahteva povratnu vrednost

public function __set($propertyName, $propertyValue) { // (do whatever needs to be done to set the property value) }

__call()

- Tu je da rukuje sa pozivima nepostojećim metodima u klasi.

- Metod prihvata ime nepostojećeg metoda kao string, i bilo koje argumente koji se prosleđuju nepostojećem metodu i to kao niz.

- Metod bi trebao da vrati vrednost nazad pozivajucem kodu

public function __call($methodName, $arguments) { // (do stuff here) return $returnVal; }

- Kreira omotač klasu koja ne sadrži svoje funkcionalnosti, samo rukuje pozivima metoda eksternih funcija ili API-a za procesovanje.

Drugi Overloading metodi:

- **__isset()** - pozvan kad god pozivajući kod pokuša da pozove PHP-ovu isset() funkciju na nevidljivom svojstvu
- **__unset() -** pozvan kad pozivajući metod pokuša da obriše nevidljiv property sa PHP unset() funkcijom
- **__callStatic() -** radi kao __call, samo što je pozvan kad god je pokušano pozvati nevidljivu statičku metodu

Nasleđivanje

- Klasa dete nasledjuje sva svojstva i metode svoje roditeljske klase, takođe može da doda još properties i metoda

class Shape { // (General Shape properties and methods here) }

class Circle **extends** Shape { // (Circle-specific properties and methods here) }

Overriding metode

```
class ParentClass {
        public function someMethod() { // (do stuff here) } }
class ChildClass extends ParentClass {
         public function someMethod() { // This method is called instead for ChildClass objects } }
 $parentObj = new ParentClass;
 $parentObj- > someMethod(); // Calls ParentClass::someMethod()
 $childObj = new ChildClass;
$childObj- > someMethod(); // Calls ChildClass::someMethod()
```

Pozivanje funkcionalnosti roditeljske klase

-Override metod roditeljske klase u klasi deteta, ali želimo da koristimo neke funkcionalnosti koje su u metodi roditeljske klase.

parent::someMethod();

```
class Banana extends Fruit {
        public function consume() {
                echo " < p > I'm breaking off a banana... < /p > ";
                parent::consume(); } }
```

Blokiranje nasleđivanja sa finalnim klasama

```
final class HandsOffThisClass {
        public $someProperty = 123;
         public function someMethod() { echo "A method"; } }
// Generates an error: // "Class ChildClass may not inherit from final class (HandsOffThisClass)"
 class ChildClass extends HandsOffThisClass { }
```

Blokiranje Override sa finalnim metodima

```
class ParentClass {
        public $someProperty = 123;
        public final function handsOffThisMethod() { echo "A method"; } }
// Generates an error: // "Cannot override final method
ParentClass::handsOffThisMethod()"
 class ChildClass extends ParentClass {
        public function handsOffThisMethod() { echo "Trying to override the method"; } }
```

Apstraktne klasei metodi

abstract class MyClass {

abstract public function myMethod($param1, $param2); }

- Ne možemo instancirati apstraktnu klasu

- Svako dete ove klase mora da implementira apstraktne metode koji su u apstraktnoj klasi, sem ako te child klase same već nisu deklarisane kao apstraktne.

- Unutar apstraktne klase možemo imati i obične metode i apstraktne.

Intrerfejsi

-Klasa može da implementira više od jednog interfejsa.

- Interfejs ne sme da sadrži property, mogu samo da imaju deklaracije metoda.

- Svi metodi interfejsa moraju biti public

interface MyInterface {

public function myMethod1($param1, $param2);

public function myMethod2($param1, $param2); }

class MyClass implements MyInterface {...

Konstruktori:

- Klasa može imati samo jedan konstruktor.

- Možemo proslediti argumente konstruktoru.

Child klasa poziva konstruktor roditeljske klase parent::__construct()

class MyClass {

function **__construct()** { echo “Whoa! I’ve come into being. < br / > ”; } }

$obj = new MyClass; // Displays “Whoa! I’ve come into being.”

Destruktori

- Poziva se pred samo brisanje objekta:

- kada su sve reference na njega nestale

- kada se završi skripta, da li prirodno ili kad nastane error

- Destruktor ne prihvata argumente

function **__destruct()** { // (Clean up here) }

Učitavanje Class fajlova

- Dobro je čuvati klase u odvojenim script fajlovima, jedna klasa po fajlu.

- Dobro je imenovati klas fajlove po klasama koje sadrže

```
<?php
require_once( “classes/Person.php” );
$p = new Person();
?>
```

- require_once() - dozvoljava da uključiš jedan PHP script fajl unutar drugog

- Razbiti PHP aplikacije u manje i lakse upravljive script fajlove

Storing Objects as Strings

```
class Person {
  public $age;
}
$harry = new Person();
$harry->age = 28;
$harryString = serialize( $harry );
echo "Harry is now serialized in the following string:
  '$harryString' <br /> ";
$obj = unserialize( $harryString );
echo "Harry's age is: $obj->age <br /> ";

Harry is now serialized in the following string:
  'O:6:"Person":1:{s:3:"age";i:28;}'
```

Before serialization PHP attempts to call a method with the name __sleep()

__wakeup() method that is called when the object is unserialized

Određivanje klase objekta

get_class – vraća string

a imamo i **instanceof** operator

II GRUPA

8. Forme

Uvod:

- Mogućnost da se prihvati ulaz od osobe koja koristi aplikaciju.

- Do sad, sve skripte koje su kreirane nisu dozvoljavale običan user input.

- Najčešći način da se dobije input od korisnika Web aplikacija je putem HTML formi

- Primer:

- Kontak forme koje omogućavaju da se pošalje email vlasnik sajta
- Forme porudžbine za poručivanje sa neke online prodavnice…

HTML fomre

-Kolekcija HTML elemenata umetnuta unutar standarne web stranice

- Različiti tipovi elemenata: text fields, pull – down meni, check box

```
< form action="myscript.php" method="post" >
  < !-- Contents of the form go here -- >
< /form >
```

Atributi forme:

- **action –** govori web pretraživaču gde da pošalje podatke forme koju popuni korisnik i submituje

  - apsolutni URL kao što je http://www.example.com/myscript.php

- relatinvi URL kao što je myscript.php ili /myscript.php…

- Skripta u navedenom URL trebalo bi da može da prihvati i procesuje podatke forme

- **method –** govori pretraživaču **kako** da pošalje podatke forme
  - **GET:**
    - Dodaje podatke forme na URL i to u name/value parovima
    - Dužina URL-a je ograničena(oko 3000 karaktera)
    - nikada ne koristiti GET metodu da se pošalju osetljivi podaci(jer će biti vidljivi u URL-u)
    - korisno sa slanje formi gde korisnik hoće da bookmark-uje rezultat
    - GET je bolja sa non-secure podatke kao što su query stringovi u google
  - **POST:**
    - Dodaje podatke sa forme unutar tela HTTP zahteva(podaci nisu vidljivi u URL)
    - Nema ograničenja veličine
    - Submissions sa POST metodom ne mogu biti bookmarkovane

Hvatanje podataka sa forme

- Atribut action sa forme mora da ima url od PHP skripte koji će rukovati sa formom

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

- Kada korisnik pošalje formu, njegovi podaci se salju serveru i form_handler.php skript je pokrenut.

- Tada skripta mora da čita podatke sa forme i da deluje po tome.

- Da bi se pročitali podaci sa forme, koriste se superglobalne promenljive.

Superglobal Array	Description
$_GET	Contains a list of all the field names and values sent by a form using the get method
$_POST	Contains a list of all the field names and values sent by a form using the post method
$_REQUEST	Contains the values of both the $_GET and $_POST arrays combined, along with the values of the $_COOKIE superglobal array

- Svaki od ova tri superglobalna niza sadrži ime polja sa poslate forme kao ključ niza sa poljem vrednosti koji su dati kao vrednosti niza.

Npr. forma koja koristi get method i ta forma sadrži sledeću kontrolu.

```
<input type="text" name="emailAddress" value="" />
```

- Pritupiti vrednosti koju je korisnik uneo u to polje forme ili sa $_GET ili sa $_REQUEST

```
$email = $_GET["emailAddress"];
$email = $_REQUEST["emailAddress"];
```

Problemi sa bezbednosti

- Nikada nećete prikazati lozinku koju je korisnik uneo(možete poslati mail da ih podsetite)

- Nikada ne treba verovati korisničkom inputu na javnom web sajtu

- Nikada prosleđivati ono što je korisnik uneo direktno iskazu kao što je echo() ili print().

-Uvke validirati ili filtrirati korinsički ulaz da bi obezbedili da je sve bezbedno pre nego što ga prikažete na web stranici

Prazna poljaa forme

- Često korisnici zaborave ili ne žele da popune neko polje u formi

- Ponekad podaci nisu poslati serveru, ponekad je polje poslato kao prazan string, ponekad ni ime polja nije poslato.

Form Control	What Happens When It ' s Not Filled In Or Selected
Text input field	The field name is sent, along with an empty value.
Password field	The field name is sent, along with an empty value.
Checkbox field	Nothing is sent at all.
Radio button field	Nothing is sent at all.
Submit button	Nothing is sent at all if the button isn ' t clicked. This can happen if the user presses Enter/Return to submit a form. However, if there ' s only one submit button in the form, most browsers will still send the button ' s field name and value.
Reset button	Nothing is ever sent.
File select field	The field name is sent, along with an empty value.
Hidden field	The field name is sent, along with an empty value.
Image field	Same behavior as a submit button.
Push button	Nothing is ever sent.
Pull - down menu	Impossible to select no option, so a value is always sent.
List box	Nothing is sent at all.
Multi - select box	Nothing is sent at all.
Text area field	The field name is sent, along with an empty value.

Kada ništa nije poslato za neko polje, PHP ne kreira element za to polje u $_POST, $_GET, $_REQUEST nizu

- pokušaj da se pristupi elementu će generisati PHP notice

PHP Notice: Undefined index: gender in process_registration.php on line 18

- Proveriti postojanje poslatog polja forme pre nego što se koristi sa onim isset npr

```
<dt>Gender</dt>
<dd><?php if (isset($_POST["gender"])) echo $_POST["gender"]?>
</dd>
```

Multi – Value polja

-Multi – select list box i checkbox sa istim imenom

- Dodati uglaste zagrade posle polja imena u HTML formi

- PHP kreira ugnježden niz vrednosti unutar $_GET $_POST $_REQUEST superglobalnog niza, umesto jedne vrednosti

```
<select name="favoriteWidgets[]" id="favoriteWidgets" size="3"
multiple="multiple">
  <option value="superWidget">The SuperWidget</option>
  <option value="megaWidget">The MegaWidget</option>
  <option value="wonderWidget">The WonderWidget</option>
</select>
```

-Pokupiti niz koji sadrži poslato polje vrednosti kao:

```
$favoriteWidgetValuesArray = $_GET["favoriteWidgets"];
$favoriteWidgetValuesArray = $_POST["favoriteWidgets"];
```

Remember: check the existence of these variables:

```
if isset($favoriteWidgetValuesArray)
  ...
```

Generisanje formi sa PHP-om

-Moguće je kombinovati obe forme i form handler unutar istog php fajla

Prednosti:

- Ako korisnici nisu popunili formu korektno, možete da ponovo prikažete formu kako bi ispravili grešku
- možete dinamički postavljati razne delove forme kada je pokrenuta skripta, dodavajući dosta moći i fleksibilnosti formi

Čuvanje PHP promenljivih u formi

- Sakrivena polja su specijalni tipovi input elemenata koji mogu da skladište i šalju string vrednosti kao i regularni text input kontrole.

- Sakrivena polja ništa ne prikazuju na stranici(iako se njihova vrednost vidi kada se pregleda page source)

- Njihova vrednost se ne može promeniti od strane korisnika kada popunjavaju formu.

- Ima mogućnost da skladišti podatke između jednosg browser request-a i sledećeg.

-Bezbednost: Iako korisnici ne mogu da menjaju vrednost hidden polja kada koriste pretraživač unutar normalnih slučajeva, veoma je jednostavno nekom napadaču da prosledi dormu koja sadrži promenjene vrednosti hidden polja. Prema ttome, nije dobra ideja da se oni koriste za prenošenje osetljivih ili kritičnih informacija, barem ne bez izvršavanja dodatne validacije u skripti da se obezbedi da su dostavljeni podaci ispravni.

File upload

```
<input type="file" name="fileSelectField"
  id="fileSelectField" value="" />
```

- Forma koja sadrži polje za selekciju fajlova mora da koristi post metod i mora imati **enctype="multipart/form-data"** atribut u svom form tagu

```
<form action="form_handler.php" method="post"
  enctype="multipart/form-data">
```

Možemo imati proizvoljan broj file select polja.

Pristupanje informacijama u upload-ovanom fajlu

-PHP kreira superglobalni niz $_FILRS koji sadrži razne informacije o fajlu-fajlovima.

-Svaki fajl je opisan jednim elementom u $_FILES nizu čiji je ključ ime polja koje se koristilo da se fajl upload-uje.

```
<input type="file" name="photo" value="" />
$_FILES["photo"]
$filename = $_FILES["photo"]["name"];
```

Elementi $_FILES niza:

Array Element	Description
name	The filename of the uploaded file.
type	The MIME type of the uploaded file. For example, a JPEG image would probably have a MIME type of image/jpeg , whereas a QuickTime movie file would have a MIME type of video/quicktime .
size	The size of the uploaded file, in bytes.
tmp_name	The full path to the temporary file on the server that contains the uploaded file. (All uploaded files are stored as temporary files until they are needed.)
error	The error or status code associated with the file upload.

error element

Constant	Value	Meaning
UPLOAD_ERR_OK	0	The file was uploaded successfully.
UPLOAD_ERR_INI_SIZE	1	The file is bigger than the allowed file size specified in the upload_max_filesize directive in the php.ini file.
UPLOAD_ERR_FORM_SIZE	2	The file is bigger than the allowed file size specified in the MAX_FILE_SIZE directive in the form.
UPLOAD_ERR_NO_FILE	4	No file was uploaded.
UPLOAD_ERR_NO_TMP_DIR	6	PHP doesn't have access to a temporary folder on the server to store the file.
UPLOAD_ERR_CANT_WRITE	7	The file couldn't be written to the server's hard disk for some reason.
UPLOAD_ERR_EXTENSION	8	The file upload was stopped by one of the currently loaded PHP extensions.

Ograničavanje veličine File upload

. Dobra je ideja da se spreči da se veoma veliki fajlovi šalju na server

- U php.ini fajl

```
; Maximum allowed size for uploaded files.
upload max filesize = 32M
```

- dodati skriveno polje forme zvano MAX_FILE_SIZE pre file upload polja

```
<input type="hidden" name="MAX_FILE_SIZE" value="10000" />
<input type="file" name="fileSelectField"
  id="fileSelectField" value="" />
```

- proveriti veličinu postavljenog fajla rulno i odbiti ako je prevelik

```
if ( $_FILES["photo"]["size"] > 10000 ) die( "File too big!" );
```

Čuvanje i korišćenje postavljenih fajlova

- Kada je fajl uspešno postavljen, automatski se čuva u privremenom folderu na serveru.

- Da bi se koristio fajl, potrebno je premestiti ga sa funkcijom move_uploaded_file() koja ima dva argumenta: pitanju fajla koji se premešta i putanju na koju će se premestiti.

. Funkcija vraća true ako je bilo uspešno premeštanje, ako se desila greška onda false.

```
if ( move_uploaded_file( $_FILES["photo"]["tmp_name"],
  "/home/matt/photos/photo.jpg" ) ) {
  echo "Your file was successfully uploaded.";
} else {
  echo "There was a problem uploading your file – please
  try again.";
}
```

Preusmeravanje posle slanja forme

- URL preusmeravanje se često koristi kod rukovanja sa kodom forme.

- Uglavnom skripta uradi svoju stvar, pokaže neku vrstu odgovora na stranici i izađe

- Slanjem specijalnih http headera naza pretraživaču sa PHP skripte, može prouzrokovati da pretraživač skoči na novi url kada je skripta pokrenuta.

- Preusmeriti korisnika na thank you sttranicu nakon što su proslali formu, tu stranicu držati nezavnisno od php skripte što omogućava stranici da se lakše edituje i ažurira.

- Ovo onemogućava korsiniku da slučajno ponovo pošalje resubmituje formu tako što će pritisnuti Reload ili refresh dugme na pretraživaču.

header("Location: thanks.html");

- **Bitno:** nikad ne prikazivati sadržaj na pretraživaču da li to bilo preko echo ili print ili uključivanjem HTML markup izvan <?php..?> pre poziva header()

- Kada se pošalje sadržaj pretraživaču, php engine automatski šalje default HTTp header koji neće sadržati location: header - i vi možete poslati headere samo jednom po zahtevu.

```
<?php
if ( isset( $_POST["submitButton"] ) ) {
  // (deal with the submitted fields here)
  header( "Location: thanks.html" );
  exit;
} else {
  displayForm();
}
function displayForm() {
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
  lang="en">
<head>
. . .
```

9. Query String-ovi

- Ranije su skripte bile samostalne, svaki zahtev koji browser uputi veb serveru je nezavisan od prethodnih zahteva (kada veb server primi zahtev da obradi PHP skriptu, on učita skriptu u svoju memoriju, pokrene je i izbriše sve njene tragove iz svoje memorije). Veb aplikacije koje se koriste danas imaju potrebu da skladište podatke između zahteva browser-a (npr. na online prodavnicama treba da se zapamti koji su artikli dodati u korpu, forumska aplikacija mora da zapamti identitet korisnika svaki put kad on postavi poruku u forumu). Rešenje za ovaj problem su:

- **Query stringovi** – za skladištenje male količine podataka u URL
- **Kolačići (Cookies)** – za skladištenje veće količine podataka u sam browser
- **Sesije (Sessions) –** za skladištenje još veće količine podataka i to na mnogo sigurniji način

- Dakle, **query stringovi** predstavljaju brz i zgodan način za prosleđivanje malih količina podataka između zahteva browsera.

Primeri: pamćenje ključnih reči koje je korisnik uneo prilikom korišćenja funkcije pretrage, identifikovati koju temu u okviru foruma treba prikazati korisniku, odrediti koju objavu na blogu treba prikazati.

- Problem je u tome što korisnik može veoma lako izmeniti podatke o stringu upita (query string data) jer su vidljivi i mogu se menjati u traci za adresu pretraživača (browser's address bar). Zato bi se trebali koristiti samo u situacijama gde slanje pogrešnih podataka neće ugroziti bezbednost. Query strings ne treba da se koriste za stvari poput korisnikovog ID-a. Ne treba se oslanjati na query stringove za autentifikaciju korisnika jer ljudi obično šalju URL-ove prijateljima u mejlovima ili aplikacijama za razmenu poruka.

http://localhost/myscript.php?firstName=Fred&lastName=Bishop&...

- Query string je deo URL-a iza znaka ?. Sadrži "name=value" parove gde je svaki par odvojen sa &

- Query stringovi nisu ograničeni na podatke obrasca. To je samo string karaktera koji se nalaze u URL-u. Zato možemo ručno kreirati URL koji sadrži query string unutar PHP skripte pa uključiti URL kao link na prikazanoj stranici ili u mejlu.

(*ne možemo samo staviti & kad ih tako pišemo već ga moramo zapisivati kao & zbog HTML-a*)

Primer:

```
$firstName = “John”;

$age = “34”;

$queryString = “firstName=$firstName&amp;amp;age=$age”;

echo ‘<p> <a href=”moreinfo.php?’ . $queryString . ‘”> Find out more info on this person </a> </p>’;
```

Ovaj kod generiše sledeći HTML kod:

```
<p> <a href=“moreinfo.php?firstName=John&amp;amp;age=34“> Find out more info on this person </a> </p>
```

Kada korisnik klikne na taj link, pokrenuće se *moreinfo.php* i query string podaci (firstName=John & age=34) će biti prosleđeni *moreinfo.php* skripti

- Dozvoljeni su samo sledeći simboli: slova, brojevi, simboli -, _, ., !, ~, *, ‘ i , drugi karakteri poput razmaka, {} ili ? nisu dozvoljeni jer koriste URL encoding. Kodira sve rezervisane znakove kao heksadecimalne brojeve kojima prethodi simbol % sa izuzetkom znaka razmak koji je kodiran kao znak +. Znakovi koji ne treba da se kodiraju (slova i brojevi) se šalju takvi kakvi jesu. Za ovo se koristi funkcija **urlencode()** kojoj prosleđujemo string za kodiranje i ona vraća kodirani string. Da bi dekodirali neki URL-encoded string koristimo funkciju **urldecode().**

- Postoji i funkcija **http_build_query()** koja uzima asocijativni niz imena polja i vrednosti i vraća čitav query string. Dodatno možemo proslediti string PHP-ovoj **htmlspecialchars()** funkciji koja konvertuje npr. & u & automatski.

- Da bi smo pristupili podacima u query stringovima potrebno je samo pročitati ih iz **$_GET** superglobalnog niza, primer korišćenja:

```
<?php

    $firstName = $_GET[“firstName”];

    $homePage = $_GET[“homePage”];

    $favoriteSport = $_GET[“favoriteSport”];

    echo “ <dl> ”;    //dl je za Description List u HTML-u

    echo “ <dt> First name: </dt> <dd> $firstName </dd> ”; //dt je description term a dd description data

    echo “ <dt> Home page: </dt> <dd> $homePage </dd> ”;

    echo “ <dt> Favorite sport: </dt> <dd> $favoriteSport </dd> ”;

    echo “ </dl> ”;

?>
```

10. Cookies

-Cookie – dozvoljavaju da skladištimo male količine podataka(ne veći od 4KB) unar korisnikovog pretraživača.

- Kad god pretraživač napravi zahtev za stranicu na vašem Web sajtu, svi podaci iz cookie se automatski šalju serveru sa tim zahtevom

- Jednom pošalješ podatke pretraživaču, ti podaci su automatski dostupni vašpj skripti od tog momenta pa nadalje.

- Mogu da traju fiksno vreme ili mogu da isteknu kada je aplikacija na pretraživaču zatvorena.

- Većina modernih pretraživača može da skladišti do 30 cookie-a po Web sajt domenu.

- Dodatno, lako je isključiti podršku za cookie u većini pretraživača.

Cookie komponente

- Kolačići se šalju sa servera na pretraživač kao deo HTTP zaglavlja.

Set-Cookie: fontSize=3; expires=Tuesday, 6-Jan-2009 17:53:08 GMT; path=/; domain=.example.com; HttpOnly

Delovi:

- **name –** ime samog cookie-a
- **value –** vrednost cookie-a
- **expires –** kada se dosegne vrednost, cookie je obrisan sa pretraživača i više se ne šalje nazad severu pri zahtevu. Ako je ova vrednost postavljena na nula ili izostavljena, cookie živi sve dok je pretraživač pokrenut, automatski se briše kad se ugasi.
- **Path –** putanja na koju bi pretraživač trebao da pošalje sam cookie nazad. Ako je navedeno, pretraživač će slati cookie samo na URL-ove koje sadrže ovu putanju, ako nije pretpostavlja se da je trenutni direktorijum skripte. Korišćenjem vrednosti „/“ govorimo da želimo da cookie bude dostupan za sve URL-ove na vašem web sajtu
- **domain –** podrazumevano je da pretraživač salje nazad cookie tačno onom računaru koji ga je i poslao. Postavljanjem ovog polja, pretraživač će slati cookie nazad na sve URL-ove unutar tog domena
- **secure –** specificira da se cookie treba slati samo ako je pretraživač napravio bezbednu (https) konekciju sa serverom
- **HttpOnly** – ako je prisutno, govori pretraživaču da bi trebao da napravi podatke cookie-a pristupačne samo skriptama koje su pokrenute na web serveru (to je preko HTTP). Pokušaji da se pristupi cookie-u preko javaScript-a unutar web stranice su odbijeni.

Bezbednost

- Više su bezbedni od query stringova- pretraživač će po default-u samo slati cookie nazad web sajtu koji ga je kreirao.

- I dalje lako za napadače da razbiju.

- Ne bi trebalo da se oslanjamo na podatke u kolačićima samo za indentifikaciju i autentifikaciju korisnika.

- I ako možemo da koristimo **domain** polje da dobijemo da pretraživač šalje cookies nazad samo mašinama unutar istog domena, ne možemo koristiti ovaj trik da postavljamo cookie za slanje na druge domene

Npr. Ako vaš web sajt www.example.com pokuša da postavi cookie sa domain vrednosti od www.google.com, taj cookie će biti odbijen od pretraživača

Postavljanje Cookie

- Direktno sa Set-Cookie: HTTP zaglavlje(korišćenjem PHP header() funkcije)

- Jednostavnije : korišćenjem ugrađene funkcije setcookie(), koja može poslati odgovarajuć HTTP header da se kreira cookie u pretraživaču

setcookie() funkcija

- Prihvata argumente za svako polje unutar cookie-ja i to po gore prikazanom redosledu.

- Samo name argument je obavezan

- Navedite bar name, value, expires i path kako biste izbegli bilo kakvu nejasnoću.ž

- Argument expires treba da bude u UNIX timestampformatu (br sec između ponoći 1. januara 1970. i željenog trenutka)

Pozovite setcookie() pre nego što pošaljete bilo koji izlaz pregledaču, jer setcookie() treba da pošalje Set-Cookie: HTTP zaglavlje.

Primer setcookie(“fontSize”, 3, time() + 60 * 60 * 24 * 365, “/”, “.example.com”, false, true);

setcookie(“pageViews”, 7, 0, “/”, “”, false, true);

Pristupanje cookie-ju

- Čitanje vrednosti iz $_COOKIE superglobal niza

- Ovaj asocijativni niz sadrži listu svih vrednosti cookie poslatih od pretraživača u trenutnom zahtevu, ključ je cookie name.

echo $_COOKIE[“pageViews”]; // Displays “8”

- Ne bi trebalo direktno prikazivati podatke iz $_COOKIE niza bez prethodnog filtriranja i/ili validacije.

- Nedavno kreirani cookie nije dostupan skriptama preko $_COOKIE dok se ne izvrši sledeći zahtev pretraživača

- Prvi put kada se skripta pokrene, samo šalje cookie pretraživaču. Pretraživač ne vraća cookie serveru dok ne zatraži URL od servera.

setcookie(“pageViews”, 7, 0, “/”, “”, false, true);

echo isset($_COOKIE[“pageViews”])

Ne vraća ništa, samo false, kad se ponovo učita onda vrati true.

- Slično, ako ažurirate vrednost cookie-a, niz $_COOKIE još uvek sadrži staru vrednost tokom izvršavanja skripta.

Uklanjane cookie-a

- Da bi se uklonio cookie, pozvati setcookie() sa imenom cookie i bilo kojaa vrednost, kao što je prazan string, i proslediti za argument za expires neko vreme koje je bilo u prošlosti.

- setcookie(“fontSize”, “”, time() - 3600, “/”, “.example.com”, false, true);

- Kada se cookie briše sa setcookie(), on se ne briše iz $_COOKIe niza sve dok je skripta pokrenuta

11. Sesije

- Cookie imaju probleme poput:

- nisu veoma bezbedni – napadač može lako da izmeni sadržaj cookie-a
- svi podaci u cookie za neki web sajt su poslati svaki put kada pretraživač pošalje zahtev za URL na serveru

- Oba problema mogu da se prevaziđu uz pomoć PHP sesije

- PHP sesija čuva podatke na serveru, vezuje kratak session ID sttring (SID) sa tim podacima
- PHP mašina(engine) tada šalje cookie koji sadrži SID pretraživaču da skladišti
- Kada pretraživač pošalje zahtev za URL na web sajtu, šalje SID cookie nazad serveru, dozvoljavajući PHP-u da donese podatke sesije i napravi ih dostupno skripti.

- ID sesije su generisani od PHP-a su jedinstveni, nasumični i skoro nemogući za pogoditi.

- Zbog činjenice da se podaci sesije čuvaju na serveru, ne moraju se slati sa svakim zahtevom pretraživača. Skladišti više podataka sesija od cookie-a

Skladištenje podataka

- PHP skladišti podatke svake sesiju u privremenom fajlu na serveru.

- Lokacija privremenog fajla je specificirana od strane session.save_path direktive na PHP konfiguracionom fajlu.

echo ini_get(“session.save_path”);

- Treba imati u vidu da sesije su napravljene samo da bi čuvale privremene podatke vezane za trenutnu interakciju korisnika.

- Cookie sesije su namešteni da isteknu kada se pretraživač ugasi.

Kreiranje sesije:

- Da se započne PHP sesija u skriptu potrebno je pozvati session_start() funkciju.

- Ukoliko je nova sesija, ova funkcija kreira jedinstveni SID za nju i šalje ga pretraživaču kao cookie koji se zove **PHPSESSID.**

- Ukoliko pretraživač pošalje PHPSESSID cookie serveru jer sesija već postoji, onda funkcija session_start() koristi postojeću sesiju.

- Potrebno je o pozvati session_start() pre bilo kog outputa ka pretraživaču.

Korišćenje podataka sesije

- Skladištenje svih podataka sesije kao ključevi i vrednosti $_SESSION[] superglobal nizu.

Npr.

$_SESSION[“firstName”] = “John”;

echo($_SESSION[“firstName”]);

$userDetails = array(“firstName” => “John”, “lastName” => “Smith”, “age” => 34);

$_SESSION[“userDetails”] = $userDetails;

Korišćenje podataka sesije i objekti

- Ako čuvamo objekte koji uključuju i definiciju klasa ili class definition fajlove, pre samog dobavljanja objekata iz $_SESSION niza

```
session_start();
class WebUser {
  public $firstName;
  public $lastName;
}
if ( isset( $_SESSION[“user”] ) ) {
  // Make sure the WebUser class is defined by this point
  print_r( $_SESSION[“user”] );
} else {
  echo “Creating user...”;
  $user = new WebUser;
  $user-> firstName = “John”;
  $user-> lastName = “Smith”;
  $_SESSION[“user”] = $user;
```

Uništavanje sesije

- Podrazumevano je da sesije automatski brišu kada korisnik isključi pretraživač, zato šro PHPSESSID cookie expires polje je stavljeno na 0.

- Ponekad je neophodno da se uništi sesija odmah(npr kada se korisnik izloguje)

- To se radi pozivom na session_destroy() funkcije, što uništava sesiju sa diska

- Podaci su idalje na $_SESSION nizu sve dok se ne završi skripta.

$_SESSION = array(); session_destroy();

- Čak i posle toga sesija postoji i dalje u obliku PHPSESSID cookie u korisnikovom pretraživaču.

- Kada korisnik poseti vaš sajt, PHP je pokupiti PHPSESSID cookie i rekreiraće sesiju samo što sesija neće imati više ikakvih podataka.

if (isset($_COOKIE[session_name()])) {

setcookie(session_name(), “”, time()-3600, “/”); }

$_SESSION = array();

session_destroy();

- Moguće je raditi sa više od jedne sesije u istoj skripti korišćenjem session_name() da se kreiraju različito imenovane sesije.

Prosleđivanje ID-a sesije Query stringovima

- Kao što je rečeno ID php sesije je sačuvan u cookie.

-Šta se dešava ako je korisnik isključio cookie u pretraživaču.

- Potrebno je proslediti session ID untar linkova između stranica sajta.

- Postoji ugrađena SID kontanta

- Ako pretraživač podržava cookie, kontanta je prazna
- ako session cookie ne može biti setovan u pretraživaču, SID sadrži string

PHPSESSID=b8306b025a76a250f0428fc0efd20a11

```
<?php session_start()?>
<a href=”myscript.php?<?php echo SID; ?> ”> Home page </a>
```

- Ukoliko id sesije je uspešno sačuvan u cookie pretraživača, kod će ispisati:

```
<a href=”myscript.php?”> Home page </a>
```

- Ukoliko php ne može da kreira cookie sesije, izlaz je sledeći

```
<a
  href=”myscript.php?PHPSESSID=5bf28931309ba166b3a3ea8b67f
  f1c57”> Home page </a>
```

- Kada korisnik klikne na link da vidi myscript.php, PHPSESSID query string vrednost se automatski pokupi od strane php engine i podaci sesije su tada dostupni skripti.

- Prosleđivanje ID-a sesije kroz URL bi trebalo da se izbegava.Npr Posetilac može da pošalje link u mejlu – taj link može da sadrži ID sesije, i taj kome je poslao onda ima pristup sesiji.

Prosleđivanje ID-a sesije – drugi način:

- pokupti trenutni ID sesije korišćenjem session_id() funkcije

- umetnuti session ID u skriveno polje PHPSESSID u formu

-sesija se može propagirati kroz slanje obrazaca - formi

12. PHP i MySQL

Povezivanje sa MySQL iz PHP

- **mysqli(MySQL improved) –** ova ekstenzija je vezana za MySQL, omogućava kompletan pristup MySQL iz PHP-a. Sadrži o proceduralne(orijentisane na funkcije) i objektno orjentisane interfejse.

- **PDO(PHP Data Objects) –** Jedna objektno orjentisana ekstenzija koja je između MySQL servera i PHP engine. Jendostavan čist skup klasa i metoda. Ista ekstenzija može biti korišćenja kada se priča i o drugim sistemima baza podataka.

Pravljenje konkcije

- Kreirati novi PDO objekat sa tri argumenta:

- DSN – koji opisuje na koju bazu se povezuje
- username – ime korisnika sa kojim želite da se povežete
- password – lozinka korisnika

- Vraćeni PDO objekat služi vašoj skripti kao konekcija na bazu

$conn = new PDO($dsn, $username, $password);

- DSN . Database Source Name – string koji opisuje atribute konekcije kao što su:

- tip sistema baze podataka
- lokacija baze
- ime baze

  $dsn = “mysql:host=localhost;dbname=mydatabase”;

  $username = “root”;

  $password = “mypass”; $conn = new PDO($dsn, $username, $password);

Zatvaranje konekcije

- Iako PHP engine uglavnom zatvara konekciju kada se skripta završi. Dobro je zatvoriti konekciju ekplicitno da bi bili sigurni.

- Da bi zatvorili konekciiju, potrebno je dodeliti null promenljivoj za bazu. Ovo efektivno uništava PDO objeka: $conn = null;

Handling errors

- Veoma opisni PDOExcetion objekti.

- Da bi se postavio PDO da podiže izuzetke kad god dđe do grešaka u bazi potrebno je koristiti PDO::SetAttribute method da se postavi PDO objekatov error mode.

```
$conn = new PDO( $dsn, $username, $password );

$conn -> setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );

try {

        $conn = new PDO( $dsn, $username, $password );

        $conn- > setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );

} catch ( PDOException $e ) {

        echo “Connection failed: “ . $e -> getMessage(); }
```

U realnim aplikacijama, potrebno je logovati sve poruke grešaka u jedan fajl, i poslati email Webmasteru kako bi ga informisali o detaljima greške.

Čitanje podataka

- Da bi se poslao SQL iskaze(statemente) MySql serveru potrebno je koristiti query metod PDO objekta:

```
        $conn -> query ( $sql );
```

- Ako SQL iskaz vraća redove podataka kao skup rezultata, možete uzeti te podatke tako što dodelimo rezultat promenljivoj.

```
$rows = $conn -> query ( $sql );
```

- Rezultat vraćen od $conn → query je ustvari drugi tip objekta – PDOSTatement object

- Koristiti ovaj objekat zajedno sa foreach petljom da bi se prolazilo kroz redove result set-a

- Svaki red je asocijativni niz koji sadrži sva imena polja i vrednosti za taj red u tabeli.

```
$sql = “SELECT * FROM fruit”;

 $rows = $conn -> query( $sql );

foreach ( $rows as $row ) {

echo “name = “ . $row[“name”] . “<br/>
”; echo “color = “ . $row[“color”] . “ <br/>
”; }
```

SQL

- Koristiti sve SQL mogućnosti:

- Ograničiti broj redova koji su vraćeni
- Sortirati vraćene redove u bilo kom redosledu
- Koristiti pattern matching
- Sumirati vraćene podatke

- Eliminisanje duplikata redova
- Grupisanje podataka zajedno
- Uzimanje podataka iz više tabela
- Korišćenje aliasa

Unošenje podataka

INSERT INTO table **VALUES (** value1 , value2 , ...); or

INSERT INTO table (field1 , field2 , ...) **VALUES** (value1 , value2 , ...);

- Unošenje lozinke člana putem enkripcije se radi pozivom MySQL-ove password() funkcije

- Enkriptovan string lozinke koji je vraćen tom funkcijom je dugačak 41 karaktera.

Inserting Records - Binding

```
$id = 8;
$username = "derek";
$password = "mypass";
$firstName = "Derek";
$lastName = "Winter";
$joinDate = "2008-06-25";
$gender = "m";
$favoriteGenre = "crime";
$emailAddress = "derek@example.com";
$otherInterests = "Watching TV, motor racing";
$sql = "INSERT INTO members VALUES ( :id, :username,
  password(:password), :firstName, :lastName,
  :joinDate, :gender, :favoriteGenre, :emailAddress,
  :otherInterests )";
```

```
try {
  $st = $conn -> prepare( $sql );
  $st -> bindValue( ":id", $id, PDO::PARAM_INT );
  $st -> bindValue( ":username", $username, PDO::PARAM_STR );
  $st -> bindValue( ":password", $password, PDO::PARAM_STR );
  $st -> bindValue( ":firstName", $firstName, PDO::PARAM_STR );
  $st -> bindValue( ":lastName", $lastName, PDO::PARAM_STR );
  $st -> bindValue( ":joinDate", $joinDate, PDO::PARAM_STR );
  $st -> bindValue( ":gender", $gender, PDO::PARAM_STR );
  $st -> bindValue( ":favoriteGenre", $favoriteGenre, PDO::PARAM_STR );
  $st -> bindValue( ":emailAddress", $emailAddress, PDO::PARAM_STR );
  $st -> bindValue( ":otherInterests", $otherInterests, PDO::PARAM_STR );
  $st -> execute();
} catch ( PDOException $e ) {
  echo "Query failed: " . $e -> getMessage();
}
```

Update podataka:

-Korišćenjem PDO::query() ili PDO::prepare() za prosleđivanje realnih vrednosti UPDATE statement-u

```
$id = 8;
$newEmailAddress = "derek.winter@example.com";
$sql = "UPDATE members SET emailAddress = :emailAddress WHERE id
  = :id";
try {
  $st = $conn -> prepare( $sql );
  $st -> bindValue( ":id", $id, PDO::PARAM_INT );
  $st -> bindValue( ":emailAddress", $newEmailAddress,
  PDO::PARAM_STR );
  $st -> execute();
} catch ( PDOException $e ) {
  echo "Query failed: " . $e -> getMessage();
}
```

Brisanje podataka

-Korišćenjem PDO:: query() ili PDO::prepare()

```
$id = 8;
$sql = "DELETE FROM members WHERE id = :id";
try {
  $st = $conn -> prepare( $sql );
  $st -> bindValue( ":id", $id, PDO::PARAM_INT );
  $st -> execute();
} catch ( PDOException $e ) {
  echo "Query failed: " . $e -> getMessage();
}
```