ПРОГРАМИРАЊЕ ВЕБ АПЛИКАЦИЈА

Nastavna jedinica: PREDAVANJE 5

- Kolačići
- Sesije
- OOP PHP: Klase, objekti, svojstva, metode, native metode
- Nasleđivanje
- Provera podataka: Filteri i regularni izrazi

KOLAČIĆI - COOKIES

- Kolačići služe za korisničku identifikaciju serveru
- Kolačići su male datoteke koje server upisuje na korisnički računar
- Svaki put kada se šalje zahtev serveru, zajedno se šalje i kolačić
- U PHP-u se može postaviti i pročitati vrednost kolačića
- Funkcija ***setcookie()*** postavlja kolačić
- Postavlja se pre <html> taga
- Sintaksa:

  setcookie(name, value, expire, path, domain);

KOLAČIĆI - PRIMER

- U sledećem primeru kreira se kolačić sa imenom "user" i pridružuje mu se vrednost "Alex Porter". Kolačić traje sat vremena, posle čega nestaje

```php
<?php
    setcookie("user", "Alex Porter", time()+3600);
?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 5</title>
</head>
<body>
<h1>Predavanje 5</h1>
</body>
</html>
```

- Vrednost kolačića je kodovana prilikom slanja i automatski dekodovana po prijemu – kodovanje se izbegava korišćenjem funkcije **setrawcookie()**

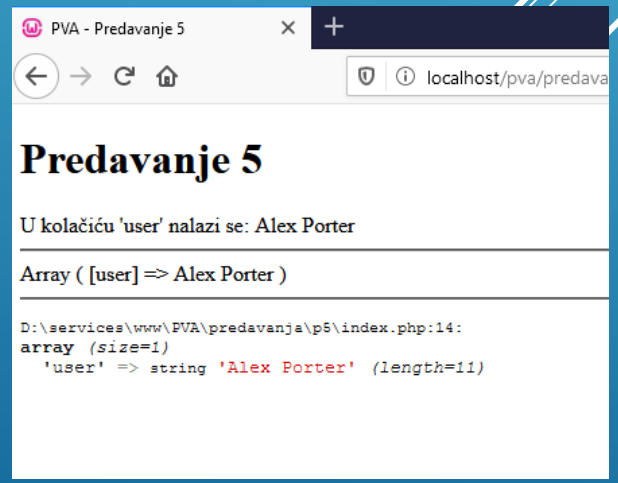
ČITANJE VREDNOSTI KOLAČIĆA

- Promenljiva **$_COOKIE** koristi se za čitanje vrednosti kolačića

```php
<?php
 echo "U kolačiću 'user' nalazi se: ".$_COOKIE['user'];
?>
```

- Sve kolačiće možemo pregledati sa ***print_r($_COOKIE)*** ili ***var_dump($_COOKIE)***

```php
<?php
 echo "U kolačiću 'user' nalazi se: ".$_COOKIE['user']."<br>";
 echo "<hr>";
 print_r($_COOKIE);
 echo "<hr>";
 var_dump($_COOKIE);
?>
```



PROVERA DA LI KOLAČIĆ POSTOJI

- Ako koristimo kolačiće na sajtu, potrebno je proveriti da li kolačić postoji pre čitanja inače PHP interpreter izbaci upozorenje.
- Za proveru koristimo funkciju ***isset()***

```
<?php
if(isset($_COOKIE['user']))
    echo "Dobro došli, ".$_COOKIE['user'];
else
    echo "Dobro došli kao gost";
?>
```

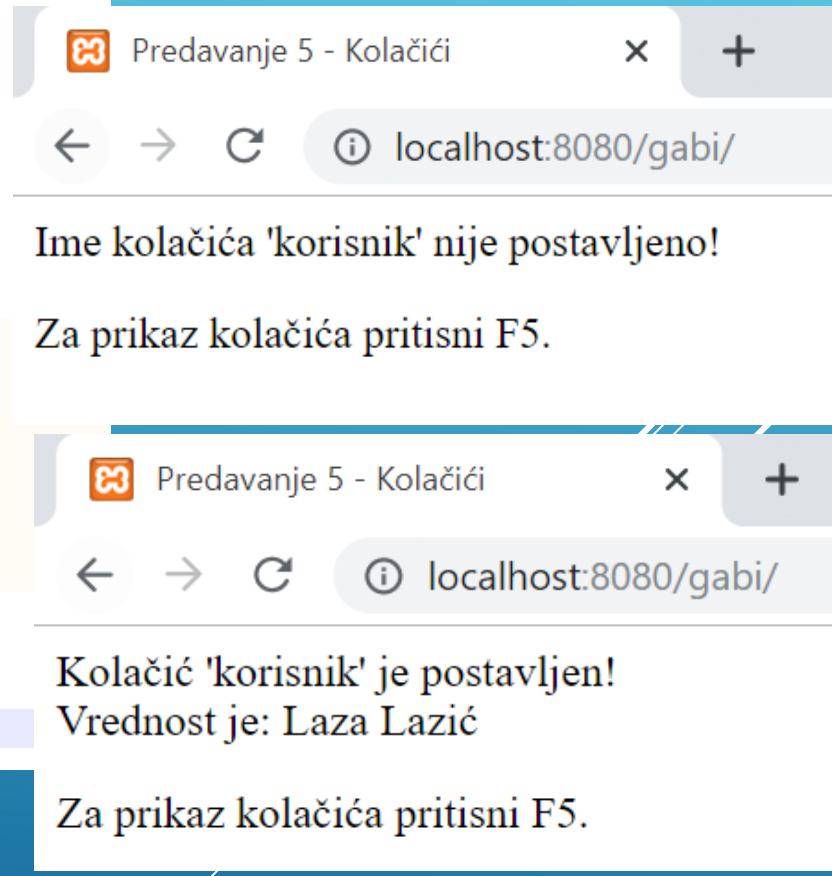
BRISANJE KOLAČIĆA

- Ne postoji instrukcija za brisanje kolačića.
- Da bi kolačić nestao dovoljno je da koristimo istu funkciju kao i za kreiranje, samo vreme trajanja postavimo u prošlosti.

```
<?php
    setcookie('user','', time()-1);
?>
```

PRIMER

```php
<!DOCTYPE html>
<?php
$cookie_name = "korisnik";
$cookie_value = "Laza Lazić";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 dan
?>
<html>
<head>
<title>
Predavanje 5 - Kolačići
</title>
</head>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Ime kolačića '" . $cookie_name . "' nije postavljeno!";
} else {
    echo "Kolačić '" . $cookie_name . "' je postavljen!<br>";
    echo "Vrednost je: " . $_COOKIE[$cookie_name];
}
?>
<p>Za prikaz kolačića pritisni F5.</p>
</body>
</html>
```



SESIJE

- Osnovna ideja upravljanja sesijama jeste da se omogući praćenje određenog korisnika tokom njegove posete nekoj Web lokaciji.
- Sesijske promenljive čuvaju podatke o korisniku na način da su vidljive svim stranicama u aplikaciji
- Da bi server tokom komunikacije sa klijentom sve vreme mogao da ga identifikuje, na serveru se čuvaju privremeni podaci koji se uništavaju po napuštanju veb sajta
- Sesiji se dodeljuje jedinstveni identifikator koji se može smestiti na klijentskom računaru u obliku kolačića ili proslediti unutar URL-a.
- Server prepoznaje korisnika na osnovu jedinstvenog ID-a i omogućava da se određene promenljive registruju kao tzv.promenljive sesije. Sadržaj tih promenljivih se čuva na serveru.

SESIJE

- Osnovni koraci upotrebe sesija su sledeći:
  - Otvaranje sesije
  - Registrovanje promenjivih sesije
  - Upotreba promenljivih sesije
  - Poništavanje promenljivih i uništavanje sesije

POKRETANJE SESIJE – SESSION_START()

- Funkcija za pokretanje sesije – **session_start()** se navodi na početku PHP skripta.

- Funkcija za pokretanje sesije započinje novu sesiju i omogućava pristup superglobalnom nizu $_SESSION.
- Ako postoji započeta sesija, funkcija *session_start()* učitava tekuće vrednosti registrovanih promenljivih sesije kako bi mogle da se koriste u skriptu.

```
<?php
    session_start();
?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 5</title>
</head>
<body>
<h1>Predavanje 5</h1>

</body>
</html>
```

PROMENLJIVE SESIJE - $_SESSION()

- Za dodelu i vraćanje sesijskih podataka koristi se promenljiva $_SESSION

```php
<?php
    session_start();
    $_SESSION['views']=1;
?>
```

- Upotreba promenljivih sesije realizuje se preko elemenata superglobalnog niza $_SESSION.

- Primer praćenja pristupa veb stranici

```php
<?php
    session_start();
?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 5</title>
</head>
<body>
<h1>Predavanje 5</h1>
<?php
if(isset($_SESSION['views']))
    $_SESSION['views']=$_SESSION['views']+1;
else
    $_SESSION['views']=1;
echo "Broj pregleda: ". $_SESSION['views'];
?>
</body>
</html>
```

УНИШТАВАЊЕ СЕСИЈЕ – SESSION_DESTROY()

- Ако желимо да прекинемо сесију, користимо функцију **session_destroy()**
- Морају се уништити у све промењиве сесије пре уништавања сесије са **unset()**

```php
<?php
    unset($_SESSION['views']);
    session_destroy();
?>
```

- Када се заврши сесија прво треба поништити све промењиве сесије, а затим позвати функцију session_destroy() како би се ослободио и идентификатор сесије.

PRIMER LOGOVANJA KORIŠĆENJEM SESIJE I KOLAČIĆA

- Ako želimo da koristimo sesije i kolačiće zajedno u okviru administrativne aplikacije moramo voditi računa i o promenljivama sesije i o kolačićima
- U narednom primeru prikazan je jedan od načina kako je moguće kombinovati oba slučaja
- Postoji stranica za logovanje (login.php), stranica koju administrator može da poseti (admin.php) i stranica sa funkcijama (funkcije.php)

LOGIN.PHP

- login.php je najzahtevnija stranica jer se tu korisnik prijavljuje. U ovom delu nema provere baze podataka, jer još nismo radili sa bazama. Korisničko ime i lozinka su ubačeni direktno u skript (hardcoded)
- Ova stranica omogućava proveru odjave korisnika, prijave korisnika i da li je korisnik već prijavljen putem kolačića ili sesije.

LOGIN.PHP – PHP DEO

```php
<?php
session_start();
require_once ("funkcije.php");
//Ako se korisnik odjavljuje
if(isset($_GET['odjava'])) odjava();

//Ako je korisnik već ulogovan, nema potrebe da dolazi na stranicu za logovanje već ga odmah prosleđujem dalje
proveraKolacica();
if(isset($_SESSION['id']) and isset($_SESSION['korime']) and isset($_SESSION['status'])) header("Location:
admin.php");

//Ako korisnik pokušava da se prijavi
if(isset($_POST['korime'])and isset($_POST['lozinka']))
{
    //Provera poslatih podataka
    if($_POST['korime']==„test" and $_POST['lozinka']==„test")
    {
        //Generisanje sesije, ako je korime i lozinka u redu, i prosleđivanje na stranicu
        $_SESSION['id']=1;
        $_SESSION['korime’]=„Test user";
        $_SESSION['status']="Administrator";
        //Provera da li korisnik želi da se zapamti na ovom računaru
        if(isset($_POST['prijava']))
        {
            //Ako želi, generišu se kolačići
            setcookie("id", 1, time()+60*60*24*30);
            setcookie("korime", „Test user", time()+60*60*24*30);
            setcookie("status", "Administrator", time()+60*60*24*30);
        }
        header("Location: admin.php");
    }
}
?>
```

LOGIN.PHP – HTML DEO

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Login</title>
</head>
<body>
<form action="login.php" method="post">
    <input type="text" name="korime" id="korime" placeholder="Unesite korisničko ime" /><br><br>
    <input type="password" name="lozinka" id="lozinka" placeholder="Unesite lozinku" /><br><br>
    <input type="checkbox" value="1" name="prijava" id="prijava">Zapamti me na ovom računaru<br><br>
    <input type="submit" value="Prijavi me">
</form>
</body>
</html>
```

ADMIN.PHP

```php
<?php
session_start();
require_once ("funkcije.php");
proveraKolacica();
prijava();
echo "Dobro došao, ".$_SESSION['korime'].", (".$_SESSION['status'].")<br>";
echo "<a href='login.php?odjava'>Odjava</a><br>";
echo "<h1>Administrativni deo</h1>";
```

FUNKCIJE.PHP

```php
<?php
function odjava()
{
    //Ako se korisnik odjavljuje, uništavaju se promenljive sesije, sesija i kolačići
    setcookie("id", "", time()-1);
    setcookie("korime", "", time()-1);
    setcookie("status", "", time()-1);
    unset($_SESSION['id']);
    unset($_SESSION['korime']);
    unset($_SESSION['status']);
    session_destroy();
}
function prijava()
{
    //Ako korisnik nije prijavljen odmah se prosleđuje na stranicu za logovanje
    if(!isset($_SESSION['id'])) header("Location: login.php");
}
function proveraKolacica()
{
    //Ako kolačići postoje generišu se promenljive sesije za dalji rad
    if(isset($_COOKIE['id']) and isset($_COOKIE['korime']) and isset($_COOKIE['status']))
    {
        $_SESSION['id']=$_COOKIE['id'];
        $_SESSION['korime']=$_COOKIE['korime'];
        $_SESSION['status']=$_COOKIE['status'];
    }
}
```

SLANJE ELEKTRONSKE POŠTE

- U veb aplikacijama često postoje forme za slanje poruka u vidu elektronske pošte, bilo kao poruka administratorima ili kao poruke među korisnicima.
- Pored komunikacije slanjem mejla, za veb aplikacije je važno i da se ti podaci smeštaju u bazu podataka (radi analize, statistike....)
- Slanje poruka putem elektronske pošte može biti sigurnosni propust o kome programer mora voditi računa
- Za slanje elektronske pošte koristi se ugrađena funkcija **mail()**

FUNKCIJA MAIL()

- Funkcija **mail()** služi za slanje elektronske pošte:

mail(to,subject,message,headers,parameters)

- Da bi funkcije za slanje elektronske pošte bile aktivirane, potrebno je to podesiti u konfiguracionoj php.ini datoteci
- Postoje i korisnički napravljene klase za slanje poruka elektronske pošte (najpoznatija je **phpmailer()**), ali one nisu predmet ovog kursa

ARGUMENTI FUNKCIJE MAIL()

Parametar	Opis
to	Potreban. Određuje adresu e-pošte primaoca
subject	Potreban. Određuje temu maila. Ovaj parametar ne može da sadrži znak za novi red.
message	Potreban. Specificira telo mail poruke. Svaka linija treba da bude odvojena sa znakom LF (\n). Linija teksta može maksimalno da sadrži 70 karaktera.
headers	Opcionalno. Specificira dodatna zaglavlja (headers) kao što su "From", "Cc", "Bcc„... Koji bi trebalo da budu odvojeni sa znakom CRLF (\r\n)
parameters	Opcionalno. Specificira bilo koji dodatni parametar.

PRIMER SLANJA PORUKE

```php
<?php
    $to = "someone@example.com";
    $subject = "Probni mail";
    $message = "Zdravo! Ovo je jednostavna poruka.";
    $from = "someonelse@example.com";
    $headers = "Od:" . $from;
    mail($to,$subject,$message,$headers);
    echo "Mail poslat.";
?>
```

PRIMER SLANJA PORUKE SA HTML DELOM

```
<?php
if (isset($_POST['email']))
//ako je "email" popunjen, pošalji mail
{
//slanje maila
    $email = $_POST['email'] ;
    $subject = $_POST['subject'] ;
    $message = $_POST['message'] ;
    mail("someone@example.com", $subject, $message, "From:" . $email);
    echo "<h3>Hvala što ste koristili našu formu za slanje e-pošte</h3>";
}
else
//ako "email" nije popunjen, prikaži formu
{
    echo "<form method='post' action='mail.php'>
    Email: <input name='email' type='text' placeholder='Unesite Vašu email adresu'><br><br>
    Subject: <input name='subject' type='text' placeholder='Unesite naslov poruke'><br><br>
    Message:<br>
    <textarea name='message' rows='15' cols='40'>Unesite poruku.......
    </textarea><br><br>
    <input type='submit' value='Pošalji poruku'/>
    </form>";
}
?>
```

KAKO PRETHODNI PRIMER RADI

- Prvo, proverava se da li je popunjeno email polje
- Ako nije postavljeno (recimo u slučaju prve posete stranici) prikazati HTML formu
- Ako je polje postavljeno (pošto je forma popunjena) preko forme se obavlja slanje maila
- Po popunjavanju forme i aktiviranju dugmeta submit, stranica se ponovo učitava, proverava se postavljanje email polja i obavlja se slanje maila
- Ovakav način slanja e-pošte je nesiguran

SIGURNOSNI PROBLEM

- Problem u prethodnom kodu ogleda se u mogućnosti korisnika da preko ulazne forme ubaci podatke u zaglavlje maila
- Šta se dešava ako korisnik unese u email polje forme sledeći tekst?

  someone@example.com%0ACc:person2@example.com

  %0ABcc:person3@example.com,person3@example.com,

  anotherperson4@example.com,person5@example.com

  %0ATo:person6@example.com
- Funkcija **mail()** postavlja gornji tekst u zaglavlje maila i zaglavlje sada ima dodatna Cc:, Bcc:, i To: polja. Pri aktiviranju submit dugmeta, e-mail se šalje na sve navedene adrese!
- Ovo je poznato kao **e-mail injection**.

ZAŠTITA OD NAPADA E-MAIL INJECTION

- Najbolji način zaštite od napada e-mail injection je provera ulaza
- Kod koji sledi je isti kao u primeru bez zaštite, s tim što je uključena provera (validacija) email polja u formi
- U tom cilju, koriste se PHP filtri ulaza:
  - **FILTER_SANITIZE_EMAIL** filter briše sve ilegalne karaktere iz e-mail stringa
  - **FILTER_VALIDATE_EMAIL** filter proverava validnost email adrese

ZAŠTITA OD NAPADA E-MAIL INJECTION - PRIMER

```php
<?php
function spamcheck($field)
{
    //filter_var() sanitizuje e-mail preko FILTER_SANITIZE_EMAIL
    $field=filter_var($field, FILTER_SANITIZE_EMAIL);
    //filter_var() proverava e-mail preko FILTER_VALIDATE_EMAIL
    if(filter_var($field, FILTER_VALIDATE_EMAIL)) return TRUE;
    else return FALSE;
}
//ako je "email" popunjen, pošalji mail
if (isset($_POST['email']))
{
    $mailcheck = spamcheck($_POST['email']);
{if ($mailcheck==FALSE)

        echo "Nevalidan ulaz.";
    }
    else
    {
        //slanje maila
        $email = $_POST['email'] ;
        $subject = $_POST['subject'] ;
        $message = $_POST['message'] ;
        mail("someone@example.com", $subject, $message, "From:" . $email);
        echo "<h3>Hvala što ste koristili našu formu za slanje e-pošte</h3>";
    }
}
else
//ako "email" nije popunjen, prikaži formu
{
    echo "<form method='post' action='mail.php'>
    Email: <input name='email' type='text' placeholder='Unesite Vašu email adresu'><br><br>
    Subject: <input name='subject' type='text' placeholder='Unesite naslov poruke'><br><br>
    Message:<br>
    <textarea name='message' rows='15' cols='40'>Unesite poruku.......
    </textarea><br><br>
    <input type='submit' value='Pošalji poruku'/>
    </form>";
}
?>
```

OBJEKTNO ORJENTISANO PROGRAMIRANJE - OOP

- Od verzije PHP 5 ubačeno je OOP
- OOP je je lakše za pisati i brže se izvršava.
- Proceduralno programiranje se zasniva na pisanju procedura (funkcija) koje se izvršavaju nad podacima, dok se OOP bazira na kreiranju objekata koji u sebi sadrže i podatke i funkcije
- OOP omogućava jasnu strukturu programa
- OOP omogućava da se kod ne ponavlja
- OOP omogućava da se naprave delovi aplikacije koji se mogu koristiti i u drugim projektima (reusable)

KLASE I OBJEKTI

- Klase i objekti su glavni delovi OOP.
- Klase su šabloni i na osnovu njih se kreiraju objekti.
- Objekti su instance klase.
- Može postojati više objekata iste klase.
- Objekat nasleđuje sva svojstva i ponašanje klase, ali svaki objekat može imati jedinstvene vrednosti.
- Klasa: automobil, objekat: Audi, BMW, Fiat.

DEFINICIJA KLASE

- Za definicuju klase koristimo rezervisanu reč ***class*** iza koje dolazi telo klase u vitičastim zagradama
- Ime klase bi trebalo biti intuitivno i nepisano pravilo je da počinje velikim slovom

```
<?php
class Ljudi{
    //svojstva i metode klase idu ovde
}
?>
```

SVOJSTVA KLASE

- Svojstva klase predstavljaju promenljive koje opisuju tu klasu

```php
<?php
class Ljudi{
    //svojstva klase
    public $ime;
    public $prezime;
    public $jmbg;
}
?>
```

- Svojstva klase mogu biti:
  - **public** – svojstvu se može pristupiti i iz klase i iz glavnog programa
  - **protected** – svojstvu se može pristupiti samo iz klase i klase koja nasleđuje ovu klasu
  - **private** – svojstvu se može pristupiti samo iz klase

METODE KLASE

- Metode klase se pišu kao i funkcije

```php
<?php
class Ljudi{
    //svojstva klase
    public $ime;
    public $prezime;
    public $jmbg;

    //metode klase
    function setIme($ime){
        $this->ime=$ime;
    }
}
?>
```

- Metode takođe mogu biti:
  - **public** – metodi se može pristupiti i iz klase i iz glavnog programa
  - **protected** – metodi se može pristupiti samo iz klase i klase koja nasleđuje ovu klasu
  - **private** – metodi se može pristupiti samo iz klase

KREIRANJE OBJEKTA KLASE

- Objekat klase se kreira korišćenjem rezervisane reči ***new***

```php
<?php
class Ljudi{
    //svojstva klase
    public $ime;
    protected $prezime;
    public $jmbg;

    //metode klase
    function setIme($ime){
        $this->ime=$ime;
    }
    function setPrezime($prezime){
        $this->prezime=$prezime;
    }
    function getPrezime(){
        return $this->prezime;
    }
}
$osoba=new Ljudi();
?>
```

КОРИШЋЕЊЕ СВОЈСТАВА И МЕТОДА КЛАСЕ

```php
<?php
class Ljudi{
    //svojstva klase
    public $ime;
    protected $prezime;
    public $jmbg;

    //metode klase
    function setIme($ime){
        $this->ime=$ime;
    }
    function setPrezime($prezime){
        $this->prezime=$prezime;
    }
    function getPrezime(){
        return $this->prezime;
    }
}
$osoba=new Ljudi();
$osoba->setIme("Pera");
echo $osoba->ime; //Izlaz je: Pera
$osoba->ime="Mile"; //public svojstvo klase
echo $osoba->ime; //Izlaz je: Mile
$osoba->setPrezime("Perić");
echo $osoba->getPrezime(); //Izlaz je: Perić
$osoba->prezime="Jović";//Greška: svojstvo je protected
?>
```

UGRAĐENE (NATIVE) METODE

- Klase u PHP-u imaju svoje ugrađene metode koje se ne pozivaju eksplicitno
- Imena metoda počinju sa dve donje crte „__“
- Neke od metoda:
  - __construct – konstruktor, poziva se prilikom kreiranja objekta
  - __destruct – destruktor, poziva se prilikom uništavanja objekta
  - __toString – pretvaranje u tekst, koristi ste prilikom štampanja objekta
  - __get – čitanje svojstava, koristi se prilikom čitanja vrednosti svojstava
  - __set – postavljanje vrednosti svojstava, koristi se prilikom dodele vrednosti svojstvima

UGRAĐENE (NATIVE) METODE

```php
<?php
class Ljudi{
    //svojstva klase
    private $ime;
    private $prezime;
    private $jmbg;

    //metode klase
 public function __construct($ime, $prezime, $jmbg){
     $this->ime=$ime;
     $this->prezime=$prezime;
     $this->jmbg=$jmbg;
 }
 public function __destruct(){
     echo "Objekat je uništen";
 }
 public function __toString(){
     return $this->ime." ".$this->prezime." (".$this->jmbg.")";
 }
 public function __set($name, $value){
     $this->$name=$value;
 }
 public function __get($name){
     return $this->$name;
 }
}
?>
```

UGRAĐENE (NATIVE) METODE

```php
<?php
$osoba=new Ljudi("Pera", "Perić", "111111111"); //Poziva se __construct metoda
echo $osoba; //Izlaz je: Pera Perić (111111111) - poziva se __toString metoda
$osoba->ime="Jovan";//Ovo je moguće zahvaljujući __set metodi
echo $osoba->ime;//Izlaz je: Pera (Ovo je moguće iako je $ime private zahvaljujući
__get metodi)
echo $osoba;//Izlaz je: Jovan Perić (111111111)
unset($osoba);//Poziva se __destruct metoda
?>
```

KORISNIČKE METODE

```php
<?php
class Kalkulator{
    private $x;
    private $y;
    public function __construct($x, $y){
        $this->x=$x;
        $this->y=$y;
    }
    public function __toString(){
        return "Klasa za sabiranje, oduzimanje, množenje i deljenje dva broja";
    }
    function sabiranje(){
        return $this->x+$this->y;
    }
    function oduzimanje(){
        return $this->x-$this->y;
    }
    function mnozenje(){
        return $this->x*$this->y;
    }
    function daljenje(){
        if($this->y!=0)return $this->x/$this->y;
        else return "Nedozvoljeno deljenje nulom";
    }
}
$calc=new Kalkulator(4, 6);
echo $calc;//Izlaz je: Klasa za sabiranje, oduzimanje, množenje i deljenje dva broja
echo $calc->sabiranje();//Izlaz je: 10
echo $calc->mnozenje();//Izlaz je: 24
$calc=new Kalkulator(5, 0);
echo $calc->daljenje();//Izlaz je: Nedozvoljeno deljenje nulom
?>
```

NASLEĐIVANJE KLASA

- PHP ima mogućnost nasleđivanja klasa
- Za nasleđivanje klasa koristi se rezervisana reč ***extends***
- Nasleđuju se sva svojstva i metode originalne klase
- Metode u roditeljskoj klasi mogu biti prepisane (overwrite) definisanjem istoimene metode u novoj klasi

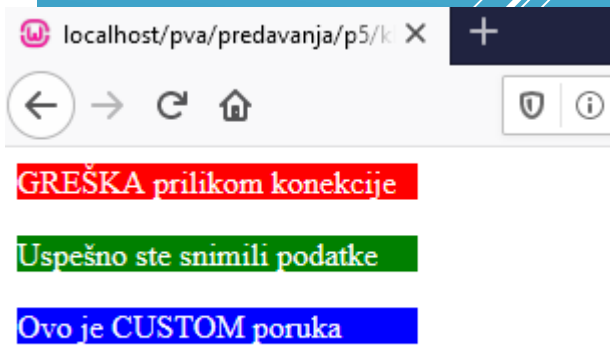
```php
class Ljudi{
    //svojstva klase
    private $ime;
    private $prezime;
    private $jmbg;

    //metode klase
    public function __construct($ime, $prezime, $jmbg){
        $this->ime=$ime;
        $this->prezime=$prezime;
        $this->jmbg=$jmbg;
    }
}

class Radnici extends Ljudi{
    public $brRadneKnjizice;
    public $firma;
    public function __construct($ime, $prezime, $jmbg, $brRadneKnjizice, $firma){
        $this->ime=$ime;
        $this->prezime=$prezime;
        $this->jmbg=$jmbg;
        $this->brRadneKnjizice=$brRadneKnjizice;
        $this->firma=$firma;
    }
    public function __toString(){
        return "Radnik: $this->ime $this->prezime $this->brRadneKnjizice ($this->firma)";
    }
}
```

```php
class Studenti extends Ljudi{
    public $brIndeksa;
    public $skola;
    public function __construct($ime, $prezime, $jmbg, $brIndeksa, $skola){
        $this->ime=$ime;
        $this->prezime=$prezime;
        $this->jmbg=$jmbg;
        $this->brIndeksa=$brIndeksa;
        $this->skola=$skola;

//ako su svojstva roditeljske klase definisana sa public ili protected opsegom može se
koristiti i parent construct umesto navođenja svojstava ime, prezime, jmbg:
                parent::__construct($ime,$prezime,$jmbg);//

    }
    public function __toString(){
        return "Student: $this->ime $this->prezime $this->brIndeksa ($this->skola)";
    }
}
$student=new Studenti("Pera", "Perić", "111111111", "RT-15/15", "VISER");
echo $student;//Izlaz je: Student: Pera Perić 123456 (VISER)
$radnik=new Radnici("Jelena", "Jovanović", "222222222", 123456, "Komercijalna banka");
echo $radnik;//Izlaz je: Radnik: Jelena Jovanović 123456 (Komercijalna banka)
```

STATIČKE METODE

- U okviru klase možemo deklarisati i statičke metode
- Za poziv statičke metode u klasi nije potrebno kreirati objekat
- Statička metoda se poziva sa duplom dvotačkom „::“

```php
<?php
class Poruke{
    public static function greskaKonekcija(){
        echo "<div style='background-color:red;color:white;width:200px'>GREŠKA prilikom konekcije</div>";
    }
    public static function uspesnoSnimanje(){
        echo "<div style='background-color:green;color:white;width:200px'>Uspešno ste snimili podatke</div>";
    }
    public static function customPoruka($poruka){
        echo "<div style='background-color:blue;color:white;width:200px'>$poruka</div>";
    }
}
Poruke::greskaKonekcija();
Poruke::uspesnoSnimanje();
Poruke::customPoruka("Ovo je CUSTOM poruka");
?>
```



STATIČKA SVOJSTVA

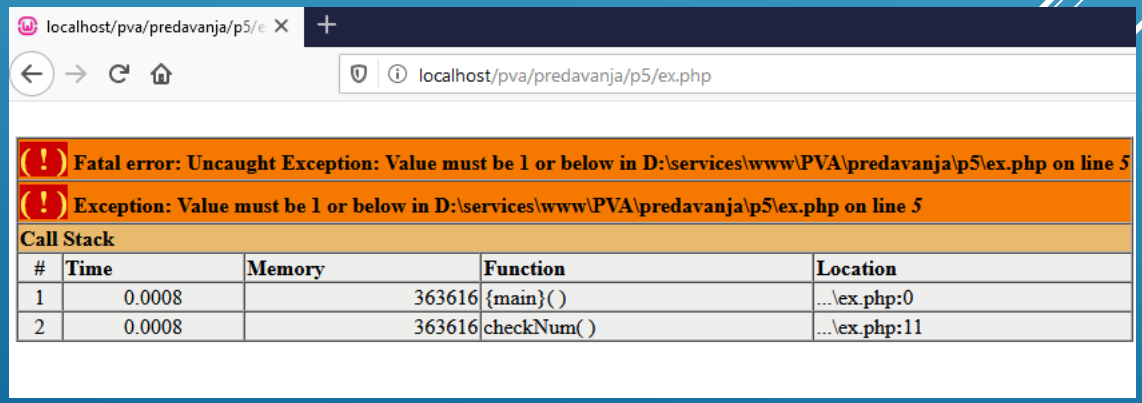
- U okviru klase možemo deklarisati i statička svojstva
- Za poziv statičkog svojstva u klasi nije potrebno kreirati objekat
- Statičko svojstvo se poziva sa duplom dvotačkom „::“

```php
<?php
class pi {
    public static $value = 3.14159;
}

echo pi::$value;//Izlaz je: 3.14159
?>
```

IZUZECI - EXCEPTIONS

- Izuzeci se koriste da izmene normalan tok programa ako se pojavi specifična greška
- Od verzije PHP5 postoji posebna klasa za obradu izuzetaka (**Exception**)
- Obrada izuzetaka se koristi da se obezbedi normalan tok izvršavanja koda ako se dođe do neželjenog stanja
- Kada se pojavi izuzetak, kod u nastavku skripta se neće izvršiti PHP će pokušati da nađe deo koda koji služi za zaobilazak greške (**catch** blok)
- Ako taj deo koda ne postoji, prikazuje se izuzetak

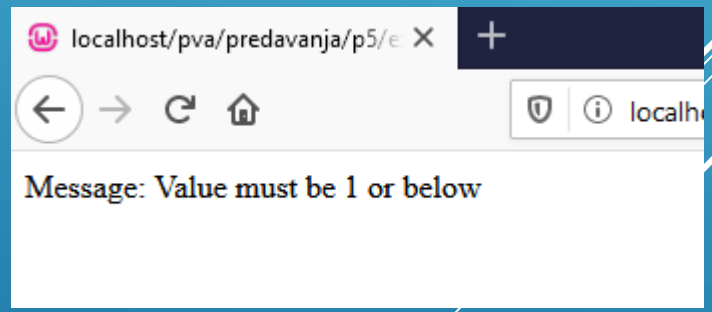
```php
<?php
//funkcija koja baca izuzetak
function checkNum($number) {
    if($number>1) {
        throw new Exception("Value must be 1 or below");
    }
    return true;
}
//generisanje izuzetka
checkNum(2);
?>
```



TRY....THROW.....CATCH

- Da bi se izbegla greška iz prethodnog primera, potrebno je napisati deo koda koji će obrađivati izuzetke
- Obrada treba da se sastoji iz **try** i **catch** blokova
- Svaki **throw** mora da ima svoj **catch**

```php
<?php
//funkcija sa izuzetkom
function checkNum($number) {
    if($number>1) {
        throw new Exception("Value must be 1 or below");
    }
    return true;
}
//generišemo izuzetak sa try blokom
try {
    checkNum(2);
    //ako postoji izuzetak nastavak koda u try bloku se neće odraditi
    echo 'If you see this, the number is 1 or below';
}
//hvatanje izuzetka
catch(Exception $e) {
    echo 'Message: ' .$e->getMessage();
}
?>
```



TRY....THROW.....CATCH

- Prethodni kod baca izuzetak i prihvata ga:
  - Kreira se funkcija **checkNum()**, koja proverava da li je broj veći od 1. Ako jeste, baca se izuzetak
  - Poziv funkcije **checkNum()** obavlja se u "try" bloku
  - Baca se izuzetak unutar funkcije **checkNum()**
  - Blok "catch" vraća izuzetak i stvara objekat (**$e**) koji sadrži informacije o izuzetku
  - Poruka o grešci od strane objekta izuzetka vraća se pozivom funkcije **$e->getMessage()**

PROVERA PODATAKA

- Najpreciznija provera unetih podataka od strane korisnika je moguća primenom filter metoda ili regularnih izraza.
- Kada klijentska provera detektuje da je sve sa podacima u redu, tek tada treba da ih pošalje serveru.
- Tako se server štiti od mnogo zahteva od korisnika sa nepotpunim i netačnim podacima i štiti se od zlonamernih napada - sql injection.
- Kada podaci dođu do servera, obavezno se moraju ponovo proveriti, sada na serverskoj strani.
- Neko je mogao da isključi JS na klijentu i ne prođe proveru ili je mogao da podatke šalje direktno URL-om i tako opet zaobiđe klijentsku proveru.

PROVERA PODATAKA

- Zašto se onda ne kontroliše samo na serverskoj strani?
- Zato što bi onda svaki pogrešan unos od strane korisnika bio poslat serveru, da ga obradi i konstatuje nenamernu grešku. Sa klijentskim proverama, za sve to se angažuje samo računar klijenta i štiti se server.
- Zato čim podatak dođe do servera, opet se proverava pa tek ako prođe i taj oblik provera, onda se sa njima nešto dalje radi u serverskom kodu.

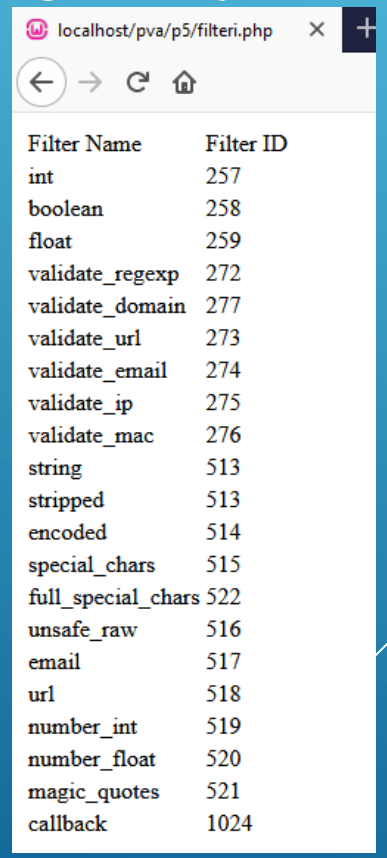
НАЧИН ПРОВЕРЕ НА СЕРВЕРУ

- Podaci se na serveru mogu proveriti na dva načina:
  - Klasično primenom regularnih izraza, na način kako je to rađeno i na klijentskoj strani samo sa drugim metodama.
  - Primenom ugrađenih funkcija PHP-a koje mogu da validiraju pojedine tipove ili formate podataka (filteri).

FILTERI

- PHP filteri služe za sanitiziranje (sanitizing) i validaciju (validating) podataka
- Sanitiziranje podataka je uklanjanje svih ilegalnih karaktera iz podatka
- Validacija podataka služi za proveru da li su podaci u odgovarajućem formatu
- U PHP postoji lista filtera koja se može koristiti (***filter_list()***)

```
<table>
    <tr>
        <td>Filter Name</td>
        <td>Filter ID</td>
    </tr>
    <?php
    foreach (filter_list() as $id =>$filter) {
        echo '<tr><td>' . $filter . '</td><td>' . filter_id($filter) . '</td></tr>';
    }
    ?>
</table>
```



Filter Name	Filter ID
int	257
boolean	258
float	259
validate_regexp	272
validate_domain	277
validate_url	273
validate_email	274
validate_ip	275
validate_mac	276
string	513
stripped	513
encoded	514
special_chars	515
full_special_chars	522
unsafe_raw	516
email	517
url	518
number_int	519
number_float	520
magic_quotes	521
callback	1024

VALIDACIJA I SANITIZACIJA

- Filtri za validaciju:
  - Koriste se za proveru tipa ili formata ulaznog podatka.
  - Zahtevaju precizna pravila formatiranja (recimo URL ili e-mail validaciju)
  - Obrada podrazumeva uklanjanje pojedinih nepoželjnih simbola ili karaktera i najčešće se koristi pre provere.
  - Vraćaju očekivani tip uspeha ili FALSE u slučaju neuspeha
- Filtri za sanitizaciju:
  - Koriste se za dozvolu ili zabranu određenih karaktera u stringu
  - Ne zahtevaju određeni format podataka
  - Uvek vraćaju string

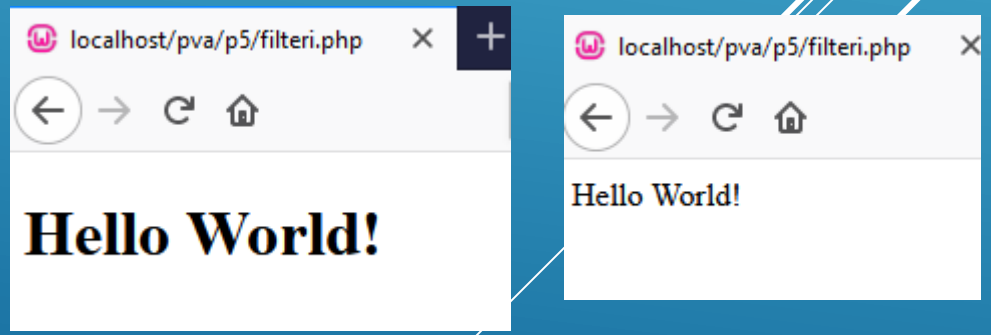
FUNKCIJE FILTERA

- Za filtriranje promenljivih koriste se sledeće funkcije:
  - filter_var() – filtrira pojedinačnu promenljivu sa određenim filtrom
  - filter_var_array() – filtrira nekoliko promenljivih sa istim ili različitim filtrima
  - filter_input() – filtrira jednu ulaznu promenljivu
  - filter_input_array() - filtrira nekoliko ulaznih promenljivih sa istim ili različitim filtrima

FILTER_VAR() FUNKCIJA

- ***filter_var()*** funkcija radi i sanitizaciju i validaciju podatka
- Ima dva ulazna parametra:
  - Promenljiva koju testiramo
  - Način testiranja
- U sledećem primeru je korišćenje filter_var() za sanitizaciju stringa. Prva slika je bez sanitizacije, druga je sa
- U ovom slučaju, funkcija se koristi za skidanje svih HTML tagova iz stringa

```php
<?php
    $str = "<h1>Hello World!</h1>";
    $str = filter_var($str, FILTER_SANITIZE_STRING);
    echo $str;
?>
```



FILTER_VAR() FUNKCIJA

- U sledećem primeru je korišćenje filter_var() za proveru integer tipa podatka
- U ovom slučaju, funkcija proverava da li je podatak zaista tipa integer

```php
<?php
    $int = 100;
    if (!filter_var($int, FILTER_VALIDATE_INT) === false)
    {
        echo("Integer is valid");
    }
    else
    {
        echo("Integer is not valid");
    }
?>
```

FILTER_VAR() FUNKCIJA

- Kada se pravi veb aplikacija važna stvar je da programer zna odakle podaci dolaze (između ostalog)
- Filter_var() se može koristiti i za proveru IP adresa sa kojih zahtevi dolaze

```php
$ip = $_SERVER['REMOTE_ADDR'];
if (!filter_var($ip, FILTER_VALIDATE_IP) === false) {
    echo("$ip is a valid IP address");
} else {
    echo("$ip is not a valid IP address");
}
?>
```

```php
<?php
$ip = "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";
if (filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6)) {
    echo("$ip is a valid IPv6 address");
} else {
    echo("$ip is not a valid IPv6 address");
}
?>
```

FILTER_VAR() FUNKCIJA

- Provera e-mail adrese prilikom unosa od strane korisnika je vrlo važna

```php
<?php
$email = "john.doe@example.com";
// Uklanjanje svih nedozvoljenih karaktera
$email = filter_var($email, FILTER_SANITIZE_EMAIL);
// Validacija e-mail adrese
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
    echo("$email is a valid email address");
} else {
    echo("$email is not a valid email address");
}
?>
```

FILTERI ZA VALIDACIJU

FILTER_VALIDATE_BOOLEAN	"boolean"	default	**FILTER_NULL_ON_FAILURE**	Returns **TRUE** for "1", "true", "on" and "yes". Returns **FALSE** otherwise. If **FILTER_NULL_ON_FAILURE** is set, **FALSE** is returned only for "0", "false", "off", "no", and "", and **NULL** is returned for all non-boolean values.
FILTER_VALIDATE_DOMAIN	"validate_domain"	default	**FILTER_FLAG_HOSTNAME**	Validates whether the domain name label lengths are valid. Validates domain names against RFC 1034, RFC 1035, RFC 952, RFC 1123, RFC 2732, RFC 2181, and RFC 1123. Optional flag **FILTER_FLAG_HOSTNAME** adds ability to specifically validate hostnames (they must start with an alphanumberic character and contain only alphanumerics or hyphens).
FILTER_VALIDATE_EMAIL	"validate_email"	default	**FILTER_FLAG_EMAIL_UNICODE**	Validates whether the value is a valid e-mail address. In general, this validates e-mail addresses against the syntax in RFC 822, with the exceptions that comments and whitespace folding and dotless domain names are not supported.

FILTERI ZA VALIDACIJU

FILTER_VALIDATE_FLOAT	"float"	default, decimal	**FILTER_FLAG_ALLOW_THOUSAND**	Validates value as float, and converts to float on success.
FILTER_VALIDATE_INT	"int"	default, min_range, max_range	**FILTER_FLAG_ALLOW_OCTAL**, **FILTER_FLAG_ALLOW_HEX**	Validates value as integer, optionally from the specified range, and converts to int on success.
FILTER_VALIDATE_IP	"validate_ip"	default	**FILTER_FLAG_IPV4**, **FILTER_FLAG_IPV6**, **FILTER_FLAG_NO_PRIV_RANGE**, **FILTER_FLAG_NO_RES_RANGE**	Validates value as IP address, optionally only IPv4 or IPv6 or not from private or reserved ranges.
FILTER_VALIDATE_MAC	"validate_mac_address"	default		Validates value as MAC address.
FILTER_VALIDATE_REGEXP	"validate_regexp"	default, regexp		Validates value against regexp, a Perl-compatible regular expression.
FILTER_VALIDATE_URL	"validate_url"	default	**FILTER_FLAG_SCHEME_REQUIRED**, **FILTER_FLAG_HOST_REQUIRED**, **FILTER_FLAG_PATH_REQUIRED**, **FILTER_FLAG_QUERY_REQUIRED**	Validates value as URL (according to » http://www.faqs.org/rfcs/rfc2396), optionally with required components. Beware a valid URL may not specify the HTTP protocol *http://* so further validation may be required to determine the URL uses an expected protocol, e.g. *ssh://* or *mailto:*. Note that the function will only find ASCII URLs to be valid; internationalized domain names (containing non-ASCII characters) will fail.

FILTERI ZA VALIDACIJU

```
if(filter_var($email, FILTER_VALIDATE_EMAIL))
 {//
 if( filter_var($value, FILTER_VALIDATE_INT) )
 {//
if (filter_var('0.0', FILTER_VALIDATE_FLOAT))
{//
 if (filter_var($url, FILTER_VALIDATE_URL))
 {//
```

FILTERI ZA SANITIZACIJU

ID	Name	Flags	Description
FILTER_SANITIZE_EMAIL	"email"		Remove all characters except letters, digits and *!#$%&'*+-=?^_`{|}~@.[]*.
FILTER_SANITIZE_ENCODED	"encoded"	FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_STRIP_BACKTICK, FILTER_FLAG_ENCODE_LOW, FILTER_FLAG_ENCODE_HIGH	URL-encode string, optionally strip or encode special characters.
FILTER_SANITIZE_MAGIC_QUOTES	"magic_quotes"		Apply addslashes().
FILTER_SANITIZE_NUMBER_FLOAT	"number_float"	FILTER_FLAG_ALLOW_FRACTION, FILTER_FLAG_ALLOW_THOUSAND, FILTER_FLAG_ALLOW_SCIENTIFIC	Remove all characters except digits, +- and optionally .,*eE*.
FILTER_SANITIZE_NUMBER_INT	"number_int"		Remove all characters except digits, plus and minus sign.
FILTER_SANITIZE_SPECIAL_CHARS	"special_chars"	FILTER_FLAG_STRIP_LOW, FILTER_FLAG_STRIP_HIGH, FILTER_FLAG_STRIP_BACKTICK, FILTER_FLAG_ENCODE_HIGH	HTML-escape '"<>& and characters with ASCII value less than 32, optionally strip or encode other special characters.
FILTER_SANITIZE_FULL_SPECIAL_CHARS	"full_special_chars"	FILTER_FLAG_NO_ENCODE_QUOTES,	Equivalent to calling htmlspecialchars() with **ENT_QUOTES** set. Encoding quotes can be disabled by setting **FILTER_FLAG_NO_ENCODE_QUOTES**. Like htmlspecialchars(), this filter is aware of the default_charset and if a sequence of bytes is detected that makes up an invalid character in the current character set then the entire string is rejected resulting in a 0-length string. When using this filter as a default filter, see the warning below about setting the default flags to 0.

FILTERI ZA SANITIZACIJU

FILTER_SANITIZE_STRING	"string"	**FILTER_FLAG_NO_ENCODE_QUOTES**, **FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_STRIP_BACKTICK**, **FILTER_FLAG_ENCODE_LOW**, **FILTER_FLAG_ENCODE_HIGH**, **FILTER_FLAG_ENCODE_AMP**	Strip tags, optionally strip or encode special characters.
FILTER_SANITIZE_STRIPPED	"stripped"		Alias of "string" filter.
FILTER_SANITIZE_URL	"url"		Remove all characters except letters, digits and $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=.
FILTER_UNSAFE_RAW	"unsafe_raw"	**FILTER_FLAG_STRIP_LOW**, **FILTER_FLAG_STRIP_HIGH**, **FILTER_FLAG_STRIP_BACKTICK**, **FILTER_FLAG_ENCODE_LOW**, **FILTER_FLAG_ENCODE_HIGH**, **FILTER_FLAG_ENCODE_AMP**	Do nothing, optionally strip or encode special characters. This filter is also aliased to **FILTER_DEFAULT**.

FILTERI ZA SANITIZACIJU

```
$text = "not a tag < 5 i nije ?";
echo filter_var ( $text, FILTER_SANITIZE_STRING); // -> not a tag

$x = filter_var($mail, FILTER_SANITIZE_EMAIL);
if (filter_var($x, FILTER_VALIDATE_EMAIL)) //
{
```

KORIŠĆENJE FLAG-OVA PRILIKOM PROVERE

- Filteri imaju i svoje opcije koje se mogu koristiti u filter_var() funkciji za specifične rezultate
- Npr. Uz filter ***FILTER_VALIDATE_IP*** može se koristiti opcija ***FILTER_FLAG_IPV6*** (ako želimo da filtriramo adresu po IPv6 protokolu)
- Uz filter ***FILTER_VALIDATE_STRING*** može ići nekoliko flagova:
  - ***FILTER_FLAG_NO_ENCODE_QUOTES*** – Ovaj indikator ne kodira znake navoda
  - ***FILTER_FLAG_STRIP_LOW*** – Uklanja karaktere sa ASCII vrednošću ispod 32
  - ***FILTER_FLAG_STRIP_HIGH*** - Uklanja karaktere sa ASCII vrednošću iznad 127
  - ***FILTER_FLAG_ENCODE_LOW*** – Kodira karaktere sa ASCII vrednošću ispod 32
  - ***FILTER_FLAG_ENCODE_HIGH*** - Kodira karaktere sa ASCII vrednošću iznad 127
  - ***FILTER_FLAG_ENCODE_AMP*** – Kodira karakter & u &

ID	Used with	Description
FILTER_FLAG_STRIP_LOW	FILTER_SANITIZE_ENCODED, FILTER_SANITIZE_SPECIAL_CHARS, FILTER_SANITIZE_STRING, FILTER_UNSAFE_RAW	Strips characters that have a numerical value <32.
FILTER_FLAG_STRIP_HIGH	FILTER_SANITIZE_ENCODED, FILTER_SANITIZE_SPECIAL_CHARS, FILTER_SANITIZE_STRING, FILTER_UNSAFE_RAW	Strips characters that have a numerical value >127.
FILTER_FLAG_STRIP_BACKTICK	FILTER_SANITIZE_ENCODED, FILTER_SANITIZE_SPECIAL_CHARS, FILTER_SANITIZE_STRING, FILTER_UNSAFE_RAW	Strips backtick characters.
FILTER_FLAG_ALLOW_FRACTION	FILTER_SANITIZE_NUMBER_FLOAT	Allows a period (.) as a fractional separator in numbers.
FILTER_FLAG_ALLOW_THOUSAND	FILTER_SANITIZE_NUMBER_FLOAT, FILTER_VALIDATE_FLOAT	Allows a comma (,) as a thousands separator in numbers.
FILTER_FLAG_ALLOW_SCIENTIFIC	FILTER_SANITIZE_NUMBER_FLOAT	Allows an *e* or *E* for scientific notation in numbers.
FILTER_FLAG_NO_ENCODE_QUOTES	FILTER_SANITIZE_STRING	If this flag is present, single (') and double (") quotes will not be encoded.
FILTER_FLAG_ENCODE_LOW	FILTER_SANITIZE_ENCODED, FILTER_SANITIZE_STRING, FILTER_SANITIZE_RAW	Encodes all characters with a numerical value <32.
FILTER_FLAG_ENCODE_HIGH	FILTER_SANITIZE_ENCODED, FILTER_SANITIZE_SPECIAL_CHARS, FILTER_SANITIZE_STRING, FILTER_SANITIZE_RAW	Encodes all characters with a numerical value >127.
FILTER_FLAG_ENCODE_AMP	FILTER_SANITIZE_STRING, FILTER_SANITIZE_RAW	Encodes ampersands (&).
FILTER_NULL_ON_FAILURE	FILTER_VALIDATE_BOOLEAN	Returns **NULL** for unrecognized boolean values.
FILTER_FLAG_ALLOW_OCTAL	FILTER_VALIDATE_INT	Regards inputs starting with a zero (*0*) as octal numbers. This only allows the succeeding digits to

PHP REGULAR EXPRESSION – RegExp-regex

- Regularni izrazi (regex ili regexp skraćeno) su specijalni tekstualni stringovi koji opisuju šablon za pretragu.

- Mogu se posmatrati kao unapređena verzija *wildcard* karaktera. Na primer, poznato je da *.txt se koristi kako bi se pronašli svi tekstualni fajlovi-regex ekvivalent toga bi bio .*\.txt .

- \^ **[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}**\$
  Ovaj izraz omogućava programeru da proveri da li je korisnik korektno uneo formatiranu email adresu, u samo jednoj liniji koda, bez obzira da li je u pitanju Perl, PHP, Java, Python .NET ili neki drugi programski jezik.

PHP REGULAR EXPRESSION – RegExp-regex

- Regularni izrazi predstavljaju način da se kontrolišu uneti stringovi u polja nekog formulara.
- Kod regularnih izraza postoji skup dozvoljenih karaktera.
- String ili neki njegov deo može imati samo one elemente koji su definisani u tom skupu.
- Regularni izrazi postoje u skoro svim "višim" programskim jezicima

PHP REGULAR EXPRESSION – PRIMER

```
if ($_POST['posalji']) {
 $ime=$_POST['ime'];
 $prezime=$_POST['prezime'];
 $brindeksa=$_POST['brindeksa'];
//Proverava da li su popunjena sva polja
  if($ime=='' || $prezime=='' || $brindeksa=='' ) {
      echo 'Sva polja moraju biti popunjena, proverite svoje podatke';
      exit;
}
if (!preg_match("[0-9]{1,3}/[0-9]{2}$",$brindeksa)) {
       echo 'Pogrešan format indeksa, ispravite unos';
       exit;
    }
else{ // neka dalja obrada podataka} }
```

PHP REGULAR EXPRESSION – RegExp-regex

PHP jezik ima ugrađene funkcije koje omogućavaju rad sa regularnim izrazima.

preg_match() – koristi za detektovanje podudaranja sa definisanim uzorkom.Ova funkcija vraća *true* ako se string *$pattern* nalazi u stringu *$string* , tj. false ako ne.

preg_split() - koristi se za detektovanje podudaranja sa definisanim uzorkom, a potom razdvajanje rezultata u numerički niz.

preg_replace() –koristi se za detektovanje podudaranja sa definisanim uzorkom i zamenu podudaranja sa određenim tekstom.

REGULAR EXPRESSION SINTAKSA

Znaci koji imaju posebno značenje kod regularnih izraza su : . * ? + [ ] ( ) { } ^ $ | \

RegExp	Značenje
[abc]	Podudaranje sa bilo kojim od karaktera a, b, ili c.
[^abc]	Podudaranje sa bilo kojim od karaktera isključujući a, b, ili c..
[a-z]	Podudaranje sa bilo kojim karakterom od a do z.
[A-Z]	Podudaranje sa bilo kojim karakterom od A do Z.
[a-Z]	Podudaranje sa bilo kojim karakterom od a do Z.
[0-9]	Podudaranje sa nekom od cifri 0 do 9.
[a-z0-9]	Podudaranje sa karaketrom između a i z ili između cifri 0 do 9.

REGULAR EXPRESSION SINTAKSA

Prečice	Značenje
.	Podudaranje sa bilo kojim karakterom osim novog reda \n.
\d	Podudaranje sa bilo kojom cifrom od 0 do 9. Isto kao [0-9]
\D	Podudaranje sa bilo karakterom da nije cifra. Isto kao [^0-9]
\s	Podudaranje sa bilo kojim karakterom koji označava belinu (space, tab, novi red ili carriage return). Isto kao [\t\n\r]
\S	Podudaranje sa bilo kojim karakterom koji nije belina. Isto kao [^ \t\n\r]
\w	Odgovara bilo kom karakteru reči definisane sa a do z, A do Z,0 do 9 i donjom crtom. Isto kao [a-zA-Z_0-9]
\W	Odgovara bilo kom karakteru koji nije definisan za reči. Isto kao [^a-zA-Z_0-9]

REGULAR EXPRESSION SINTAKSA

Simboli ? , + , * i {} označavaju broj ponavljanja definisanog karaktera u stringu:

RegExp	Značenje
p+	Jedno ili više pojavljivanja slova p.
p*	Nula ili više pojavljivanja slova p.
p?	Nula ili jedno pojavljivanje slova p.
p{2}	Tačno dva pojavljivanja slova p.
p{2,3}	Najmanje dva pojavljivanja, ali ne više od tri pojavljivanja slova p.
p{2,}	Dva ili više pojavljivanja slova p.
p{,3}	Najviše tri pojavljivanja slova p.
^p	Slovo p na početku reda.
p$	Slovo p na kraju reda.

REGULAR EXPRESSION SINTAKSA

Modifikator	Značenje
i	Ne uzima se u obzir velika i mala slova - case-insensitive
m	Menja značenje ^ i $ u skladu sa označavanjem novog reda. Npr. početak i kraj svakog reda u višelinijskom stringu umesto navođenja granica.
g	Globalno podudaranje, pronalazi sva pojavljivanja.
o	Proverava izraz samo jednom
s	Menja značenje . (tačke) u podudaranje sa svim karakterima uključujući i novi red.
x	Dozvoljava upotrebu belina i komentara unutar regularnog izraza radi jasnoće.

TAČKA

- Tačka se koristi kao džokerski znak. To znači da se može upotrebiti umesto bilo kog drugog znaka.

- “i..” tako daje mogućnost da se definiše bilo koji skup od tri znaka koji počinje sa i (ict, ips, irc, ..)

- Ako se tačka želi fizički prikazati, a ne koristiti kao džoker znak, onda se ispred nje dodaje obrnuta kosa crta “\.rs” definiše skup “.rs”

Primer:
x.y proverava da li se iza x nalazi bilo koji karakter pa y
a.*o uslovljava da je iza a bilo koji karakter(i) ili nista i da se završava sa o
("ao", "auto", "ameriko")

PRIMERI:

- ^x proverava da li neki string počinje sa x
- x$ proverava da li se neki string završava sa x
- "x" proverava da li se u nekom stringu nalazi x
- [x] proverava da li se u nekom stringu nalaze karakteri iz x
- ^.{5}$ string sa tačno pet (nekih) karaktera (škola, ulica,..)
- [^x] proverava da li se u nekom string-u ne nalaze karakteri iz x
- (b|cd)x može biti "bx" ili "cdx".

PRIMERI:

- [tu] string sadrži t ili u (pandam je "t|u") – samo jedan može biti vraćen
- m[ae]čka - može biti mačka ili mečka
- [a-d] string sadrži neko od slova a do d (pandam "a|b|c|d" ili "[abcd]")
- ^[a-zA-Z] string koji počinje slovom
- [0-9]% string koji ima jednu cifru pre znaka %
- ,[a-zA-Z0-9]$ string koji se završava sa , i alfanumeričkim znakom
- [^a-zA-Z]%" karakter između % % nije slovo

PREG_MATCH()

preg_match($pattern, $input, $matches, $flags, $offset): Vraća prvo pojavljivanje podudaranja postavljenog šablona u ulaznom stringu.

preg_match_all($pattern, $input, $matches, $flags, $offset): Vraća sva pojavljivanja podudaranja postavljenog šablona u ulaznom stringu.

- $pattern: string šablon po kome se vrši pretraživanje
- $input: ulazni string koji se pretražuje
- $matches: ako postoji podudaranje, rezultate smešta u numerički niz $matches.

PREG_MATCH()

```php
<?php
$pattern = "/ca[kf]e/";
$text = "He was eating cake in the cafe.";
if(preg_match_all($pattern, $text,$matches)){
    echo "Match found!";
    print_r($matches);
} else{
    echo "Match not found.";
}
?>
```

```
Match found!Array ( [0] => Array ( [0] => cake [1] => cafe ) )
```

PREG_MATCH()

name@somwhere.com
first.last@somewhere.com
first.last@place.someplace.us
my_name@me.info
me.something@somewhere.co.uk

Potrebno je kreirati šablon: (nešto)@ (nešto). (nešto), gde nešto može biti bilo šta sem znaka @ i praznine

```
<?php
function checkEmail($email) {
$pattern = "/^[A-z0-9\._-]+"
. "@"
. "[A-z0-9][A-z0-9-]*"
. "(\.[A-z0-9_-]+)*"
. "\.([A-z]{2,6})$/";
return preg_match ($pattern, $email);
}
?>
```

PREG_MATCH()

Password koji mora početi i završiti se slovima i mora imati više od 8 karaktera

```
<?php
function checkPassword($password) {
$length = strlen ($password);
if ($length < 8) {
return FALSE;
}
return preg_match ("/[A-z]+[0-9]+[A-z]+/", $password);
}
?>
```

PREG_MATCH()

```
preg_match("/^[0-9+\-\.\/\(\) ]{6,30}$/", $telefon);
// samo karakteri iz skupa [+/()]
```

* Dozvoljeno je pojavljivanje 6 do 30 karaktera iz skupa [0-9+\-\.\/\(\)]
- 0-9+ znači da se cifre 0 do 9 mogu pojaviti jednom ili više puta

```
//validacija datuma u mysql formatu (YYYY-MM-DD)

preg_match("/^[0-9]{4}-[0-9]{2}-[0-9]{2}$/", $datum);
```

PREG_SPLIT()

preg_split($pattern, $subject, $limit, $flag)

```php
<?php

$str  = 'ProgramiranjeVebAplikacija';

$izlaz = preg_split('//', $str , -1,PREG_SPLIT_NO_EMPTY);

print_r($izlaz);
?>
```

Array ([0] => P [1] => r [2] => o [3] => g [4] => r [5] => a [6] => m [7] => i [8] => r [9] => a [10] => n [11] => j [12] => e [13] => V [14] => e [15] => b [16] => A [17] => p [18] => l [19] => [20] => k [21] => a [22] => c [23] => i [24] => j [25] => a)

```php
<?php

$str  = 'Programiranje Veb Aplikacija';

$izlaz = preg_split("/[\s,]+/", $str);

print_r($izlaz);
?>
```

Array ([0] => Programiranje [1] => Veb [2] => Aplikacija)

PREG_REPLACE()

preg_replace($pattern, $replacement, $subject, $limit, $count)

```php
<?php
$string = 'abcde$ddfd @abcd )der]';
echo 'Ulazni string: '.$string.'';
$novi = preg_replace("/[^A-Za-z0-9 ]/", '', $string);
echo "<br>";
echo 'Novi string : '.$novi."\n";
?>
```

Ulazni string: abcde$ddfd @abcd)der]
Novi string : abcdeddfd abcd der

```php
<?php
$str1 = "$12,334.00A";
echo preg_replace("/[^0-9,.]/", "", $str1)."\n";
?>
```

12,334.00