**Napredni stringovi i paterni**

Jedinica: 14 od 19

**Rezime**

* Postoji nekoliko načina za zapis stringova i mnoštvo funkcija za rukovanje njima;
* PHP je u stanju da parsira string tako što će iz njega izvući komponente bitne za izvršenje programa (promenljive);
* Razlika između dvostrukih i jednostrukih navodnika je, između ostalog, što dvostruki navodnici dozvoljavaju parsiranje promenljivih unutar stringa, a jednostruki ne;
* Stringove je moguće formatirati po potrebi izlaza;
* Moguće je prebrojati karaktere stringa (funkcija strlen);
* Moguće je promeniti određeni deo stringa nekim drugim stringom (str\_replace);
* Moguće je izdvojiti određeni tekst u nekom stringu (substr).

U ovoj lekciji ćete naučiti napredne string funkcije i opštu manipulaciju stringovima. U prethodnoj lekciji govorili smo o stringovima. Tada smo naučili osnovne operacije koje su nam potrebne za proizvodnju bilo kakve web aplikacije. U ovoj lekciji produbićemo ta znanja i dopuniti ih novim.

Počećemo sa **interpolacijom promenljivih**.

PHP ima jednu veoma dobru osobinu – u stanju je da parsira string tako da promenljive koje se u njemu nalaze imaju različit tretman od samog teksta. Mi smo u prošloj lekciji obradili funkciju printf, koja omogućava formatiranje šeme, po kojoj će biti tretirani parametri funkcije *printf("my variable%d", $x);*

Ali, ovo smo mogli uraditi i na drugi način:

|  |  |
| --- | --- |
| 1 | printf("my variable $x"); |

odnosno:

|  |  |
| --- | --- |
| 1 | echo "my variable $x"; |

Rezultat će u oba slučaja biti: *my variable 2*(dva je u primeru proizvoljan broj. Biće, naravno, emitovana vrednost promenljive $x). PHP prilikom ovog postupka nije svemoćan. I treba paziti da promenljive koje želimo da tretiramo kao promenljive ne izgube kontekst.

Pogledajmo sledeći primer:

|  |  |
| --- | --- |
| 1 | echo "$a bcd"; |

Promenljiva $a će biti regularno emitovana, jer je razdvojena od ostatka sadržaja stringa. Ali, u slučaju da linija izgleda ovako:

|  |  |
| --- | --- |
| 1 | echo "$abcd"; |

neće biti emitovana ni promenljiva $a, ali ni ostatak teksta, jer će interpreter tražiti promenljivu pod nazivom $abcd.

Ukoliko želite da implementirate string tako da promenljiva i ostatak stringa budu spojeni, možete upotrebiti vitičaste zagrade kao separatore:

|  |  |
| --- | --- |
| 1 | echo "{$a}bcd"; |

ili konkatenacione operatore:

|  |  |
| --- | --- |
| 1 | echo $a ."bcd"; |

U oba slučaja, rezultat će biti isti: 2bcd (ponavljamo da je vrednost 2 hipotetička. Ona može biti bilo koja vrednost promenljive $a).

Ponekad ćemo poželeti da postignemo i obrnut efekat. Želećemo da promenljiva ne bude tretirana kao promenljiva, nego kao tekst (na primer, želimo da nam tekst stringa bude „$a bcd“). Da bismo ovo uradili, obratićemo malo pažnju na navodnike u PHP-u.

Videli smo da je u PHP-u moguće praviti string na dva načina: sa jednostrukim ili dvostrukim navodnicima. Ovo ćemo obično uraditi ako želimo da se u našem tekstu vide i navodnici. Obično je svejedno u koje ćemo navodnike smestiti string, sve dok se ne poklapaju sa navodnicima koje želimo da prikažemo unutar tog stringa:

|  |  |
| --- | --- |
| 1 | echo '"my" text'; |

Ovaj primer će emitovati na stranu sledeći rezultat:

          "my" text.

I obrnuto. Sledeći primer:

|  |  |
| --- | --- |
| 1 | echo "'my' text"; |

emitovaće:

'my' text

Ali će zato:

|  |  |
| --- | --- |
| 1 | echo ""my" text"; |

prijaviti grešku.

Naravno, uvek možete staviti navodnik iza escape karaktera i tako ga emitovati: \"

Vratimo se sada na [interpolaciju](https://www.link-elearning.com/linkdl/opisPojma.php?id=139111) promenljivih. Tu takođe postoji razlika kada su tipovi navodnika u pitanju. Videli smo da dvostruki navodnici dozvoljavaju parsiranje promenljive. S druge strane, jednostruki navodnici ne parsiraju ni promenljive, kao ni specijalne karaktere, pa će sve što u njih stavite biti tako i emitovano:

|  |  |
| --- | --- |
| 1 | echo '$a bcd'; |

emituje:

$a bcd

**Herodoc sintaksa**

Iako specijalni karakteri i HTML oznake omogućavaju veliku slobodu prilikom emitovanja teksta (možete napraviti formu kakvu god želite), ovo često i nije najjednostavniji način da se tekst formatira. Naročito ako želite da kopirate neku već gotovu tekstualnu formu. U ovom slučaju, mnogo je praktičnije upotrebiti **herodoc** string. Uz pomoć ove sintakse, možete formatirati čak i najkompleksnije stringove. Upotrebljava se na sledeći način.

Početak stringa obeležava se oznakom **<<<** kojoj sledi identifikator, što može biti bilo koji naziv koji ne odstupa od standardnih pravila za imenovanje promenljivih. Obično se, zbog preglednosti, [identifikator](https://www.link-elearning.com/linkdl/opisPojma.php?id=139109) piše velikim slovima. Nakon identifikatora obavezan je novi red, odakle može početi i sam tekst.

Kraj teksta se označava takođe identifikatorom, kome mora prethoditi novi red, a sledi oznaka **;**

<<<MOJTEKST  
Naslov<br>  
Sadrzaj...  
MOJTEKST;

Ovako formatiran sadržaj možete smestiti u promenljivu ili poslati direktno na izlaz:

$a=<<<MOJTEKST  
Naslov<br>  
Sadrzaj...  
MOJTEKST;

echo<<<MOJTEKST  
Naslov<br>  
Sadrzaj...  
MOJTEKST;

Herodoc stringovi su veoma moćni alati. Pokušajte da otvorite izvorni kod neke web strane i da ga kopirate u svoj herodoc string. Videćete da će kompletan kod biti obrađen bez problema ili grešaka.

S obzirom na to da omogućava memorisanje velike količine teksta, nemojte zaboraviti da postoji memorijski limit koji je PHP-u dozvoljen za skriptiranje. U slučaju da preterate sa količinom, PHP jednostavno neće biti u stanju da obradi vaš string i doći će do greške.

**Dužina stringa**

Jedna od najčešće korišćenih funkcija nad stringovima je funkcija kojom se određuje dužina stringa. Ova funkcija se zove **strlen**.

Da bismo upotrebili ovu funkciju (koja je inače vrlo jednostavna), pogledaćemo prvo od čega se uopšte sastoji string. Npr, tekst: moj string – sastoji se od karaktera. A svaki od tih karaktera je, zapravo, jedan bajt. To znači da je naš string, u stvari, ništa drugo nego jedan niz bajtova. Da li to znači da bismo, kada bismo se obratili nekom stringu kroz njegov niz, dobili kao članove tog niza karaktere našeg stringa? Da.

Na primer:

|  |  |
| --- | --- |
| 1  2 | $a = "my text";  echo $a[3]; |

posle sledećeg koda, na ekranu će biti ispisan karakter **t**, što je drugi (ako se koristi zero based indeksiranje) karakter u našem stringu.

S obzirom na to da smo ustanovili da je string jedan niz karaktera, funkciju strlen možemo tretirati kao funkciju koja broji članove tog niza.

|  |  |
| --- | --- |
| 1 | echo strlen("my text"); |

Ono što je važno da znate prilikom korišćenja ove funkcije jeste da, za razliku od nekih drugih jezika, ona nema **zero terminated** mehanizam, i kada naiđe na prazan karakter ona ne prekida automatski brojanje, nego nastavlja, sve dok postoje bilo kakve vrednosti za karaktere u promenljivoj koju broji.

To u praksi znači da će ovaj primer:

|  |  |
| --- | --- |
| 1  2 | $a="     ";  echo strlen($a); |

koji bi u nekom drugom jeziku (sa drugom funkcijom) dao rezultat 0, ovde dati rezultat 5, jer se string sastoji od pet praznih karaktera.

Evo primera kako da se rešimo spejsova u nekom tekstu:

|  |  |
| --- | --- |
| 1  2  3  4  5 | $s = "my text";  for ($i = 0; $i < strlen ($s); $i++) {  if($s[$i] != " ")  echo $s[$i];  } |

**Poređenje stringova**

Osnovni način za poređenje stringova je uz pomoć operatora poređenja.

|  |  |
| --- | --- |
| 1 | echo "my text"=="my text"; |

Ovaj način je dobar, ukoliko rukujemo sa hard kodiranim vrednostima. Ali, pogledajte sada sledeći primer:

|  |  |
| --- | --- |
| 1 | echo "1my text"==1; |

Ovaj primer, kao i malopređašnji, daje tačan rezultat, iako, naizgled, dve poređene strane nemaju nikakve bliskosti.

PHP je prvo pregledao operande i video da se radi o poređenju stringa sa int-om, što je nemoguće. Zato je implicitno konvertovao levi operand u int, što je vrednost levog operanda svelo na 1 (jedini broj koji postoji u stringu), što je konačno dovelo do poređenja 1 i 1, koje, naravno, daje tačan rezultat. Možda, na prvi pogled, ne deluje kao da bi do ovakve greške moglo doći, ali, ne zaboravite da u programima na ovakva poređenja obično dolaze dinamički kreirane vrednosti (na primer $a==$b).

Poređenje stringova možete izvršiti i na drugi način, funkcijama **strcmp** i **strcasecmp**. Ove funkcije, kao parametre, primaju dva stringa koja se porede, a kao rezultat vraćaju nulu, ako su stringovi identični, ili broj različitih karaktera.

|  |  |
| --- | --- |
| 1 | echo strcmp("my text","my text"); |

Primer emituje vrednost 0.

Razlika između strcmp i strcasecmp je u tome što funkcija strcasecmp nije osetljiva na velika i mala slova.

|  |  |
| --- | --- |
| 1  2 | echo strcmp("My text", "my text"); // rezultat je -1  echo strcasecmp("My text", "my text"); // rezultat je 0 |

Funkcije vraćaju brojeve manje od 0 ako je string1 manji od stringa2, odnosno brojeve veće od 0 ako je string1 veći od stringa2. Ukoliko su stringovi identični, biće vraćen broj nula.

**Pretraga stringa**

Postoji mogućnost traženja određenog karaktera ili sekvence karaktera unutar samog stringa. Za ovo se koristi funkcija **strpos**.

Ova funkcija prihvata, kao parametre, inicijalni string i sekvencu koju tražimo, a kao rezultat vraća početnu poziciju (indeks) tražene sekvence. Odnosno, boolean vrednost false, ukoliko tražena sekvenca ne postoji.

|  |  |
| --- | --- |
| 1 | echo strpos("my text", "ext"); |

Ovaj primer na strani ispisuje broj: 4

Takođe, ova funkcija može prihvatiti i treći, opcioni parametar, koji označava mesto od koga će pojava traženog karaktera biti uzeta u obzir.

|  |  |
| --- | --- |
| 1 | echo strpos("my textext", "ext", 5); |

U primeru se ne uzimaju u obzir pojave traženog karaktera pre petog karaktera stringa. Znači da prvo pojavljivanje neće biti uzeto u obzir. Rezultat funkcije će biti broj 7, jer se prvo sledeće pojavljivanje događa na osmom karakteru stringa.

**Izmena stringa**

U PHP-u je moguće jedan deo stringa zameniti drugim. Ovako nešto možete uvek uraditi uz pomoć do sada pomenutih funkcija, ali i korišćenjem funkcija posebno pisanih za tu namenu: str\_replace i str\_ireplace (ista funkcija neosetljiva na velika i mala slova):

|  |  |
| --- | --- |
| 1 | echo str\_replace("my", "your", "my text"); |

funkcija str\_replace prihvata tri (ili četiri) parametra. Prvi parametar je traženi deo stringa, drugi je deo koji će biti ubačen umesto traženog stringa, a treći je string na kome se vrši intervencija. Četvrti, opcioni parametar omogućava da se u neku promenljivu smesti broj pronađenih slučajeva traženog stringa.

|  |  |
| --- | --- |
| 1  2  3 | $a=0;  echo str\_replace("my", "your", "my text", $a);  echo $a; |

Ovaj primer daće vrednost jedan, pošto se reč „moj" pojavljuje samo jednom u stringu.

Na ovaj način, možete zameniti i više slučajeva u jednom stringu, uz pomoć nizova:

|  |  |
| --- | --- |
| 1  2  3 | $arr1 = array("Java", "SQL", "CSS");  $arr2 = array("PHP","MySQL","HTML");  echo str\_replace($arr1, $arr2, "I love Java",$a); |

Ovaj primer promeniće tekst u: „I love PHP*"*.

Nekada ćete želeti da izmenite string na određenoj poziciji (indeksu). Na primer, ako ste gradili neku listu na osnovu nekih podataka koje ste dobijali sekvencijalno. Lako se može desiti da ti podaci završe u sledećem obliku:

1,2,3,4,5,

jer, s obzirom da listu kreirate dinamički, ne znate koji je njen poslednji član, pa ostanete sa jednim zarezom (ili nekim drugim separatorom) viška. U tom slučaju, **substr\_replace** funkcija je odlično rešenje:

|  |  |
| --- | --- |
| 1  2 | $x = "1,2,3,4,5,";  echo substr\_replace($x, "", strlen($x)-1); |

Funkcija kao parametre prihvata string, string koji ćete umetnuti kao zamenu i poziciju odakle će početi zamena. S obzirom na to da dužinu stringa ne znamo (lista može biti 1-5 ali može biti i 1-1000), za poziciju uzimamo dužinu umanjenu za jedan (znači, poslednji element).

**Izdvajanje dela stringa**

Možete izdvojiti deo stringa sa određene pozicije funkcijom **substr**. Ova funkcija prima tri parametra: string, poziciju početka dela koji se izdvaja i, opciono, broj karaktera koji se izdvajaju.

|  |  |
| --- | --- |
| 1  2 | $x = "http://www.google.com";  echo substr($x,7); |

Primer će emitovati tekst nakon sedmog karaktera. Rezultat je www.google.com. Kada se ne unese treći parametar, preuzima se kompletan string od naznačene početne pozicije. Ukoliko se unese treći parametar, preuzima se onoliko karaktera koliko je naznačeno u trećem parametru:

|  |  |
| --- | --- |
| 1  2 | $x = "http://www.google.com";  echo substr($x,7,3); |

Rezultat primera je www.

**Formatiranje stringa**

Formatiranje stringova smo već pomenuli kroz funkciju **number\_format**. Ova funkcija omogućava raznovrsnost prilikom ispisa brojeva (pre svega decimalne notacije):

|  |  |
| --- | --- |
| 1 | echo number\_format(30.4000,3); |

Prvi parametar ove funkcije je sam broj. Drugi parametar je maksimalan broj prikazanih decimala. Izlaz ovog koda je: 30.400.

Ova funkcija može prihvatiti i dva opciona parametra (isključivo oba parametra), koja predstavljaju karakter koji će odvajati decimale broja i karakter koji će odvajati hiljade.

|  |  |
| --- | --- |
| 1 | echo number\_format(30000,3,".",","); |

Izlaz je:

30,000.000

Neke formate možete deklarisati i putem identifikacije lokalizacije, funkcijom setlocale. Za to vam je potreban naziv grupe za formatiranje i naziv lokalizacije. Sledeći primer formatiraće sve zapise grupe LC\_MONETARY, po lokalizaciji en\_US:

|  |  |
| --- | --- |
| 1 | setlocale(LC\_MONETARY, "en\_US"); ili setlocale(LC\_MONETARY, "ja\_JP"); |

**Generičko formatiranje**

S ovim načinom formatiranja već smo upoznati kroz prethodne lekcije. Funkcije su printf, sprintf i fprintf (razlike ove tri funkcije su isključivo u tipu izlaza: prva ima standardan izlaz – strana ili konzola – sprintf može vratiti rezultat, a fprintf ima izlaz u fajl).

Sintaksa formatiranja je sledeća:

|  |  |
| --- | --- |
| 1 | printf("My number: %d", 100); |

Prvi parametar ove funkcije (string „»moj broj: %d") je ono što će otići na izlaz, dok je drugi (pa i svi ostali) parametar vrednosti koja će zameniti identifikator u stringu (%d). U ovom slučaju, identifikator je d, što znači da će na izlazu biti decimalna vrednost, ali mogli smo upotrebiti i neke druge identifikatore. Na primer: %b, bi dao binarni izlaz:

|  |  |
| --- | --- |
| 1 | printf("My number: %b", 10); |

Prilikom formatiranja stringa na ovaj način niste ograničeni samo na jedan identifikator.

|  |  |
| --- | --- |
| 1 | printf("My decimal number: %d, my binary number: %b, my floating point number: %f", 10,10,10); |

Moguće je koristiti sledeće vrednosti za formatiranje stringa:

* + b – binarna reprezentacija,
  + c – prikazuje karakter koji reprezentuje ascii brojnu vrednost,
  + d – decimalna reprezentacija,
  + e – prevodi u broj sa eksponentom (printf("%e", 10); daje izlaz 1.000000e+1),
  + u – neoznačena reprezentacija,
  + f – reprezentuje broj kao broj sa pokretnim zarezom prema zadatoj lokalizaciji printf ("%3.2f", $broj),
  + F – reprezentuje broj kao broj sa pokretnim zarezom bez zadate lokalizacije,
  + o – oktalna reprezentacija broja,
  + s – string (printf("%s", "moj tekst");),
  + x – heksadecimalna reprezentacija sa malim slovima (ffffff),
  + X - heksadecimalna reprezentacija sa velikim slovima (FFFFFF).

**Regularni izrazi**

Kada jednostavno nemate uslova da rešite neku operaciju na stringu standardnim putem (kod je, na primer, prekompleksan), onda se možete poslužiti i regularnim izrazima.

Regularni [izrazi](https://www.link-elearning.com/linkdl/opisPojma.php?id=139112) su setovi pravila po kojima se neki string pretražuje.

Na primer, ako bismo želeli da budemo sigurni da je neki string zapisan u formi e-maila, mogli bismo da kažemo da za njega postoji nekoliko pravila:

* + Mora imati neki tekst na početku, bez specijalnih karaktera,
  + Mora imati karakter @ nakon početnog teksta,
  + Mora imati tekst nakon oznake @ ,
  + Zatim, mora imati tačku i tekst nakon tačke.

Ovaj opis odgovara većini e-mailova.

Pošto iz opisa zaključujemo da znamo neke osobenosti, ali ne i koji je tačno sadržaj celog stringa, ovo je pravo mesto za upotrebu regularnog izraza.

Da bi neki tekst bio tretiran kao regularni izraz, potrebno je da mu na početak i kraj stavimo delimitere. Ovi delimiteri mogu biti bilo koji karakter, ali se u praksi najčešće u tu svrhu koristi kosa crta **/**.

/moj regularni izraz/

Sve unutar delimitera predstavlja sadržaj regularnog izraza, odnosno obrazac za poređenje.

Funkcija koja poredi regularni izraz sa stringom naziva se **preg\_match**. Ova funkcija prihvata (u osnovi) dva parametra, regularni izraz i string koji se poredi, a vraća kao rezultat 1 ako se tekst poklapa sa izrazom, i 0 ako se ne poklapa. Zapravo, funkcija vraća broj. Sledeći primer vraća rezultat 1.

|  |  |
| --- | --- |
| 1 | echo preg\_match("/mytext/","mytext"); |

Ovako banalno poređenje mogli smo izvršiti i standardnim funkcijama, dok regularne izraze obično koristimo kada ne možemo da rešimo poređenje ni na jedan drugi način. Ne samo zato što su komplikovani za upotrebu već i zato što po brzini zaostaju za standardnim funkcijama.

Da bi regularni izraz imao neku funkciju, potrebno je uvesti mu još neke elemente osim delimitera.

**Metakarakteri**

Metakarakteri su delovi regularnog izraza, koji na neki način idenitifikuju određeni deo teksta.

* + . –  Označava bilo koji karakter u tekstu   
    echo preg\_match("/my.ext/","myText"); //vraća 1
  + **ˆ**– Označava početak stringa
  + **$**– Označava kraj stringa
  + **\s**– Označava spejs   
    echo preg\_match("/my\stext/","my text"); //vraća 1
  + **\d –**Označava bilo koji broj   
    echo preg\_match("/number \d/","number 5"); //vraća 1
  + **\w –**Označava reč u stringu   
    echo preg\_match("/my \w/","my text"); //vraća 1

Moguće je grupisanje više uslova za određeni deo teksta, uz pomoć uglastih zagrada:

|  |  |
| --- | --- |
| 1 | echo preg\_match("/a[bcd]e/","abe"); |

U prethodnom primeru dozvoljena je jedna od tri alternative (b, c ili d) između karaktera a i e.

Dozvoljene su i kombinacije određenih opsega karaktera sa metakarakterima. Sledeći primer podrazumeva početno slovo a, zatim slova b ili c, kojima sleduje broj.

|  |  |
| --- | --- |
| 1 | echo preg\_match("/a[bc\d]/","ab2"); //return 1 |

**Kvantifikatori**

Kvantifikatori određuju koliko će se puta određeni uslov ponoviti u regularnom izrazu.

* + Karakter može da se pojavi jednom, više puta ili da se ne pojavljuje  
    *echo preg\_match("/moj s\*tring/","moj sssstring"); // vraća 1  
    echo preg\_match("/moj s\*tring/","moj string"); // vraća 1  
    echo preg\_match("/moj s\*tring/","moj tring"); // vraća 1*
  + Karakter može da se pojavi jednom ili više puta  
    *echo preg\_match("/moj s+tring/","moj sssstring"); // vraća 1  
    echo preg\_match("/moj s+tring/","moj string"); // vraća 1  
    echo preg\_match("/moj s+tring/","moj tring"); // vraća 0*
  + Karakter može da se pojavi jednom ili nijednom na zadatoj lokaciji  
    *echo preg\_match("/moj s?tring/","moj tring"); // vraća 1  
    echo preg\_match("/moj s?tring/","moj string");  // vraća 1  
    echo preg\_match("/moj s?tring/","moj sstring");  // vraća 0*
  + {n,m} Karakter mora da se pojavi minimum n puta, a maksimum m puta.  
    echo preg\_match("/moj s{1,3}tring/","moj ssstring"); *// vraća 1*echo preg\_match("/moj s{1,3}tring/","moj sssstring"); *// vraća 0*

**Regularni izrazi unutar regularnih izraza**

Moguće je tretirati i ceo jedan izraz kao zasebnu jedinicu (karakter). Na primer, ako bismo želeli da upotrebimo kvantifikator nad nekim izrazom, a ne samo karakterom.

Oznaka za regularni izraz unutar regularnog izraza je otvorena i zatvorena mala zagrada.

|  |  |
| --- | --- |
| 1 | echo preg\_match("/my (ab.) string/","my abc string");  //return 1 |

U primeru, rekli smo da želimo da naš obrazac mora da počinje rečju **moj**, zatim mora uslediti **prazan prostor**, a zatim moraju uslediti tri karaktera ab i bilo koji karakter. Na kraju, izraz smo završili rečju **string**, kojom prethodi jedan prazan prostor.

Ovo smo mogli uraditi i bez malih zagrada, ali šta ako želimo da na kompletan obrazac (ab.) primenimo neki kvantifikator? Na primer, da mora da se ponovi jednom ili više puta (+), mogli bismo, jednostavno, izmeniti izraz:

|  |  |
| --- | --- |
| 1 | echo preg\_match("/moj (ab.)+ string/","moj abcabdabe string"); // vraca 1 |

Regularni izrazi su posebna dimenzija programiranja, i nemojte se mnogo opterećivati njima. Iako su veoma efikasni, retko da će vam zatrebati uzorak regularnog izraza koji već ne postoji. Tako da ćete veći deo rada na regularnim izrazima provesti u traženju odgovarajućeg uzorka, nego u sopstvenoj izgradnji istog.

**Šta će biti rezultat izvršavanja koda u nastavku?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?php  $s = "my text";  for ($i = 0; $i < strlen ($s); $i++) {      if($s[$i] != " ")          echo $s[$i];  }  ?> |

 mytext

 my text

 My text

 MY TEXT

**Vežba 1**

***Problem:***

U aplikaciju ulazi sledeća promenljiva:

$string = "myMail@mail.ml";

Potrebno je napraviti regularni izraz kojim će se proveriti da li je vrednost promenljive mail adresa.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?php  $string = "myMail@mail.ml";  $pattern = "/^[a-zA-Z0-9]+\@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}$/";  echo preg\_match($pattern,$string);  ?> |

Funkcija preg\_match vrši proveru poklapanja stringa sa definisanim paternom. Potrebno je samo pojasniti definisani regularni izraz. Simbolom ^ definiše se početak izraza nakon kojeg sledi niz dozvoljenih karaktera od a do z (bez obzira na velika i mala slova). Takođe, dozvoljeno je postavljati i cifre, što definišemo opsegom 0 do 9. Ovaj niz se može pojaviti više puta. Nakon toga traži se unošenje karaktera @ kome sledi već poznat niz karaktera. Takođe, zahteva se i unošenje tačke kojoj sledi niz karaktera sličan pomenutom, ali bez dozvole za unosom cifara. Ovaj poslednji niz mora imati 2 ili 3 karaktera, nakon čega sledi kraj paterna.

**Vežba 2**

***Problem:***

Data je sledeća promenljiva:

$string = "http://myPage.php?id=25&cat=18&user=34";

Potrebno je preuzeti sve parametre i smestiti ih u asocijativni niz.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <?php  $string = "http://myPage.php?id=25&cat=18&user=34";  $pars = explode("?",$string);  $pars = explode("&",$pars[1]);  $parsedPars=array();  for($i=0;$i<sizeof($pars);$i++)      {            $currentParam = explode("=",$pars[$i]);          $parsedPars[$currentParam[0]] = $currentParam[1];      }  print\_r($parsedPars);  ?> |

Kako se parematri nalaze nakon znaka pitanja u stringu koji je dat, pozivamo funkciju explode i kao separator postavljamo "?". Sada imamo niz sa dva elementa. Na poziciji označenoj indeksom 1 nalaze se parametri koji nas zanimaju. Parametri su međusobno odvojeni znakom "&" i zato vršimo još jedan explode. Nakon ovoga smo u promenljivoj $pars dobili niz sa tri elementa. Zato prolazimo kroz for petlju i izvlačimo ključeve i vrednosti. Ovo takođe vršimo funkcijom explode, ali ovoga puta kao separator postavljamo znak "=". Na kraju je potrebno samo da prikažemo rezultate na strani:

|  |  |
| --- | --- |
| 1 | print\_r($parsedPars); |

**Vežba 3**

***Problem:***

Dat je sledeći url string:

$string = "http://myDomain/home/index.php?id=25&cat=18&user=34";

Potrebno je izolovati samo domen sa folderima i nazivom strane (myDomain/home/index.php).

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php  $string = "http://myDomain/home/index.php?id=25&cat=18&user=34";  $pars = preg\_replace("/http:\/\//","",$string);  $pars = preg\_replace("/\?[a-zA-Z0-9=&]+/","",$pars);  print\_r($pars);  ?> |

Ovim zadatkom predstavljen je još jedan primer upotrebe funkcije str\_replace ali ovoga puta se koriste i regularni izrazi. U prvom pozivu funkcije se uklanja počeni deo stringa, dok se u drugom pozivu uklanja završni. Na kraju se na strani prikazuje traženi rezultat.