Web programiranje

Vežbe 11 - Uvod u regularne izraze

(preuzeto sa www.b92.net) Autor: Dragan Dinić

Bez obzira na prirodu projekta na kome radite, pre ili kasnije doći ćete u situaciju da morate manipulisati nekim tekstualnim podacima. Bilo da je u pitanju jednostavna validacija forme ili pak parsovanje podataka sa nekog live sajta, regularni izrazi (eng. Regular Expressions) su oruđe kojim morate ovladati da bi ste bili uspešni u tome.

O važnosti regularnih izraza dovoljno govori činjenica da je podrška za njih prisutna u svim modernim programskim jezicima (u Perlu su čak deo samog jezika), a možete ih koristiti i u većini naprednih tekst editora, alatima za pretragu itd.

Potpuno ovladavanje regularnim izrazima nije nešto što možete "odraditi" preko noći, šta više, tema je toliko opširna da je o njoj napisano i nekoliko knjiga. Cilj ovog tutorijala je da vas upozna sa osnovnim mogućnostima regularnih izraza kako biste bili u mogućnosti da ih efikasno koristite.

Šta su to regularni izrazi

Regularni izraz praktično predstavlja poseban skup znakova (string) u kome se odgovarajućom sintaksom (eng. pattern) taj niz upoređuje sa nekim drugim skupom znakova. Može se koristiti za pretragu unutar nekog teksta, izvlačenje određenog podstringa, validaciju (e-maila na primer) i sl. Možda sve ovo zvuči komplikovano, ali će stvar biti mnogo jasnija, čim krenemo sa nekoliko primera.

PHP podržava takozvane POSIX kao i Perl kompatibilne regularne izraze. Iako među njima postoje izvesne razlike, osnovna sintaksa je u suštini ista, tako da ćemo sintaksu predstavljenu ovde koristiti i sa posix i sa perl kompatibilnim funkcijama. Inače, iako su primeri u PHP-u, oni se uz manje izmene mogu prilagoditi i drugim programskim jezicima, jer je sintaksa regularnih izraza manje više ista.

Sintaksa Regularnih Izraza

Prva dva specijalna karaktera sa kojima ćemo početi su '^' i '$'. Oni označavaju početak, odnosno kraj stringa. Tako na primer:

"**^foo**" - proverava da li string počinje sa "foo"

"**foo$**' - proverava da li se string završava sa "foo"

Recimo ako imamo string "Mali Perica uci PHP", rečnikom PHP-a to bi izgledalo ovako:

```
<?php
$string = "Mali Perica uci PHP";
// Vraca true ako se string "Perica" nalazi u promenljivoj $string
preg_match("/Perica/", $string);

// Vraca true ako $string pocinje sa "Mali"
preg_match("/^Mali/", $string);

// Vraca true ako se $string zavrsava sa "PHP"
preg_match("/PHP$/", $string);

// Vraca true ako string sadrzi tacnu frazu "Mali Perica uci PHP"
preg_match("/^Mali Perica uci PHP$/", $string);
?>
```

Simboli '?', '+', '*' i {} označavaju broj pojavljivanja nekog karaktera u stringu:

- **?** - Karakter koji prethodi znaku '?' može se pojaviti jednom ili nijednom (za pattern "ab?" odgovaralo bi "a","ab")
- ***** - Karakter koji prethodi znaku '*' može se pojaviti nijednom ili više puta (za pattern "ab*" odgovaralo bi "a", "ab", "abb", "abbb", ...)
- **+** - Karakter koji prethodi znaku '+' može se pojaviti jedanput ili više puta (za pattern "ab+" odgovaralo bi "ab", "abb", "abbb", ...)
- **{n}** - Karakter koji prethodi znaku {n} može se pojaviti tačno n puta (za pattern "ab{3}" odgovaralo bi "abbb")
- **{n, }** - Karakter koji prethodi znaku {n, } može se pojaviti najmanje n puta (za pattern "ab{3,}" odgovaralo bi "abbb", "abbbb", "abbbb", ...)
- **{n,m}** - Karakter koji prethodi znaku {n,m} može se pojaviti n do m puta. (za pattern "ab{2,4}" odgovaralo bi "abb", "abbb", "abbbb")

Pored broja pojavljivanja, mozemo definisati i tačan skup znakova koje string sme da sadrži. Na primer:

- **'.'** - Bilo koji karakter
- **[abc]** - Samo slova a, b i c
- **[a-z]** - Sva mala slova od a do z
- **[A-Z]** - Sva velika slova od A do Z
- **[a-zA-z]** - Sva slova, mala ili velika
- **[0-9]** - Svi brojevi od 0 - 9
- **[a-zA-Z0-9]** Svi alfanumericki karakteri

Unutar zagrada [] simbol '^' koristimo kao negaciju, tako na primer, ako želimo da naš string ne sadrži brojeve koristili bi nešto poput: [^0-9].

Pored skupova znakova koje sami definišemo, postoje već predefinisani skupovi znakova, a to su:

- **[[:alnum:]]** - Bilo koji alfanumerički karakter (isto što i [a-zA-Z0-9])
- **[[:alpha:]]** - Bilo koje slovo (isto što i [a-zA-Z])
- **[[:upper:]]** - Bilo koje veliko slovo (isto što i [A-Z])
- **[[:lower:]]** - Bilo koje malo slovo (isto što i [a-z])
- **[[:blank:]]** - Tab i space karakter (isto što i [\t])
- **[[:space:]]** - Bilo koji space karakter
- **[[:digit:]]** - Bilo koji broj (isto što i [0-9])
- **[[:xdigit:]]** - Bilo koji heksadecimalan broj
- **[[:punct:]]** - Bilo koji od znakova "."'"?!;:"
- **[[:print:]]** - Svi printabilni karakteri
- **[[:graph:]]** - Svi printabilni karakteri (osim spaceova)

I poslednje ali ne i najmanje bitno, izbor od **tačno jednog** elementa iz definisanog skupa:

(string1|string2|...|stringn)

Na primer za pattern "(a|b)cde" stringovi "acde" i "bcde" bi bili odgovarajući. Takođe, zagrade možemo koristiti za pravljenje "subpatterna", poput "ba(na)+" ("bana","banana","bananana", ...).

Toliko o teoriji, a sada da vidimo kako regularne izraze upotrebiti u praksi, za recimo validaciju forme.

```
<?php

// validacija korisnickog imena
// dozvoljavamo samo korisnicko ime koje sadrzi alfanum i donju crtu
// minimum 6, max 20 karaktera
$found = preg_match("/^[a-zA-Z0-9_]{6,20}$/", $username);

if(!$found)
{
   echo "Korisnicko ime nije validno";
}

// validacija telefona
// prihvatamo samo brojeve i karaktere iz skupa [+-/()]
//obratite paznju na koriscenje escape karaktera "\" za '.', '/', '(' i ')'
$found = preg_match("/^[0-9+\-\.\/\(\) ]{6,30}$/", $phonenum);

if(!$found)
{
   echo "Telefon nije validan";
}

//validacija datuma u mysql formatu (YYYY-MM-DD)
$found = preg_match("/^[[:digit:]]{4}-[[:digit:]]{2}-[[:digit:]]{2}$/", $datum);

if(!$found)
{
   echo "Datum nije validan";
}

// validacija usa zip koda
// USA zipocode je u formatu xxxxx ili xxxxx-xxxx
// gde je x bilo koji ceo broj
// na primer 12345 ili 12345-1234

$found = preg_match("/^([0-9]{5}|([0-9]{5}-[0-9]{4}))$/", $datum);

if(!$found)
{
   echo "Zipcode nije validan";
}

// jednostavna validacija emaila
$found = preg_match("/^[a-zA-Z0-9\.\-]+@[a-zA-Z0-9\.\-]+$/", $email);

if(!$found)
{
   echo "e-mail nije validan";
}
?>
```

Primer 1

Šta će biti izlaz sledećeg PHP skripta:

```
<html>
<head>
<title>PMF Kragujevac</title>
</head>
<body>
<?php
    $str = array("mile123@pmf.kg.ac.rs",
                 "mile@server2.pmf.kg.ac.rs",
                 "mile@pmf.kg.ac.rs",
                 "mile123@firma034.co.rs",
                 "mile@firma.co.rs",
                 "mile@mail.firma.co.rs");
    $reg1 = "/^[a-z]+@[a-z]+(\.[a-z]+)*(\.ac\.rs)$/";
    $reg2 = "/^[a-z]+@[a-z]+(\.co\.rs)$/";

    print "<p>Za izraz: <b>$reg1</b></p>";
    foreach ($str as $s) {
        if (preg_match($reg1, $s))
             print "<p>string <i>$s</i> - <b>je regularan</b></p>";
        else print "<p>string <i>$s</i> - nije regularan</p>";
    }

    print "<hr>";

    print "<p>Za izraz: <b>$reg2</b></p>";
    foreach ($str as $s) {
        if (preg_match($reg2, $s))
             print "<p>string <i>$s</i> - <b>je regularan</b></p>";
        else print "<p>string <i>$s</i> - nije regularan</p>";
    }

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

(3, 5)

Primer 2

```
<html>
  <head>
    <title>PMF Kragujevac</title>
  </head>
    <?php
      $stringovi = array("www8.dobarsajt1.edu",
                         "www678.amu.ac.zu5l",
                         "www6.ailmit.net",
                          "www.euler.ni.ac.rs",
                         "mitcl.edu",
                         "www.core.amu.edu",
                         "www.znanje.edu");
       $reg = "/^(w{3}[0-9]*\.)?[a-z]{1}[a-z0-9]{1,7}((\.ac\.[a-z]{2})|(\.edu))$/i";
       $zapis = "";
       $i=0;
```

```
            foreach ($stringovi as $s){
                  $i+=1;
                  if (preg_match($reg,$s))
                        $zapis = $zapis."$i -   TRUE<br>";
                  else
                        $zapis = $zapis."$i -   FALSE<br>";
             }
             echo $zapis;
        ?>
</html>
```

(5,7)

Primer 3

```
<html>
  <head>
    <title>PMF Kragujevac</title>
  </head>
        <?php
             $stringovi = array("E234-94_evolutionVII",
                                "E35-03_skylinerII",
                                "E876-02_CIVICiX",
                                "E76-05_targaV",
                                "e1-99_FOCusXIX",
                                "e00-89_miataI");
             $reg = "/^e[0-9]{2,3}(-|#)?0[0-9]_?[A-Z]{5,7}$/i";
             $prolaz = "";

             $i=0;
             foreach ($stringovi as $s){
                  $i++;
                  if ( preg_match($reg,$s) )
                        $prolaz = $prolaz."<b> Trace $s </b><br>\n";
                  else
                        $prolaz = $prolaz."Out  $s <br>\n";
             }
             echo $prolaz;
        ?>
</html>
```

(3,4)

Pattern modifiers:

http://php.net/manual/en/reference.pcre.pattern.modifiers.php