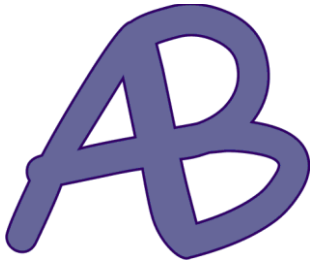


VI. PHP

- Wyrażenia regularne

dr Artur Bartoszewski
UTH Radom

Wyrażenia regularne

Wyrażenia regularne

Wyrażenia regularne umożliwiają opisywanie i przetwarzanie długich ciągów znaków.
Działają na zasadzie porównania ciągu znaków z określonym wzorem.

Dzięki wyrażeniom regularnym, możliwe jest wykonanie wielu operacji na raz np. wyszukiwanie określonych elementów, walidacja adresów email, walidacja adresów URL oraz zamiana poszczególnych fragmentów strony.

Ograniczniki

Każde wyrażenie regularne jest łańcuchem znaków i jako takie zapisywane jest w apostrofach, lecz dodatkowo powinno zostać objęte w ogranicznik na początku i końcu wyrażenia. Takim ogranicznikiem zazwyczaj jest slash (/), ale może to być również inny znak (np. @)

Uwaga: jeżeli w masce wykorzystywany jest (/) lub inny znak specjalny trzeba zapisać to tak \/ \@ itp/

Wzorzec	Dopasowania
,/abc/’	abcdef; 123abcd; qwertyabc

Zaczepienia końca i początku ciągu - ^ i $

Możemy też zażądać aby wzorzec znalazł się na początku lub na końcu ciągu.

^ - oznacza początek ciągu

$ - oznacza koniec ciągu

Wzorzec	Dopasowania
,/^abc/’	abcdef; 123abcd; qwertyabc abc
,/abc$/’	abcdef; 123abcd; qwertyabc abc
,/^abc$/’	abcdef; 123abcd; qwertyabc abc

Znaki

zakresy znaków takie jak litery A-Z, czy cyfry 0-9 są dość często używane w wyrażeniach regularnych - istnieją skrócone wersje zapisu zakresu znaków

Operator	Opis
.	Operator jednego dowolnego znaku.
[abc]	Zbiór znaków – jeden ze znaków w nawiasie
[a-z]	Dowolne litery od a do z.
[A-Z]	Dowolne litery od A do Z.
[0-9]	Dowolne cyfry od 0 do 9.
[^]	Operator negacji zbioru.
|	Operator alternatywy

Przykład:

```
’/[a-zA-Z0-9]/’          ->  dowolna litera lub cyfra
,/[^0-9]/’               ->  znak nie będący cyfrą
,/[^ąęćśżźó]             ->  znak nie będący polskim znakiem diakrytycznym
```

Znaki

Zakresy znaków takie jak litery A-Z, czy cyfry 0-9 są dość często używane w wyrażeniach regularnych - istnieją skrócone wersje zapisu zakresu znaków

```
\w - oznacza wszystkie litery, cyfry i podkreślnik - [a-zA-Z0-9_]
\W - stanowi negację \w - [^a-zA-Z0-9_]
\d - oznacza wszystkie cyfry - [0-9]
\D - stanowi negację \d - [^0-9]
\s - oznacza znaki specjalne - spację, \n, \r\ i \t
\S - stanowi negację \s
```

Przykład:

```
'/abc[^0-9]/'    <->   '/abc\D/'
```

Znaki

Zapis POSIX-owy

Wybrane zakresy można również zapisać w standardzie POSIX.

- [:alnum:] - oznacza wszystkie litery i znaki
- [:alpha:] - oznacza litery
- [:blank:] - oznacza spację lub znak tabulacji
- [:cntrl:] - oznacza znaki sterujące
- [:digit:] - oznacza cyfry
- [:graph:] - oznacza znaki drukowalne bez odstępu
- [:lower:] - oznacza jedynie małe litery
- [:print:] - oznacza znaki drukowalne ze znakiem odstępu
- [:punct:] - oznacza znaki drukowalne za wyjątkiem liter i cyfr
- [:space:] - oznacza spacje
- [:upper:] - oznacza jedynie duże litery
- [:xdigit:] - oznacza cyfry w zapisie szesnastkowym

Liczba znaków

Zwykle nie znamy dokładnej liczby znaków w wyrażeniu stąd istnienie wielu operatorów które pozwalają określić w jakim przedziale liczba ta się znajduje

Operator	Opis	Priorytet
*	Operator powtórzenia 0 lub więcej razy.	lewostronny
+	Operator powtórzenia 1 lub więcej razy.	lewostronny
?	Operator powtórzenia 1 lub 0 razy.	lewostronny
( )	Wyrażenie wewnątrz nawiasów jest **atomem** (rozpatrujemy je jako całość).	n/a
{x}	Operator powtórzenia dokładnie ‚x' razy.	lewostronny
{x,y}	Operator powtórzenia minimum ‚x' i maksimum ‚y' razy.	lewostronny
{x,}	Operator powtórzenia minimum ‚x' razy.	lewostronny
{,y}	Operator powtórzenia maksimum ‚y' razy.	lewostronny

Przykład:

```
'/[a-z]*/'       ->  mała litra wystąpi dowolną ilość razy
,/[0-9]{3}/'     ->  dokładnie trzy cyfry
,/[0-9]{,8}/'    ->  maksymalnie 8 cyfr
```

Atom

Atom – zapisywany w nawiasach ( ) - to najmniejsze wyrażenie we wzorze. Nie rozbijamy go na mniejsze składowe.

- Z atomów zbudowany jest wzór wyrażenia regularnego.
- Atomem może być pojedyncza litera lub cyfra.
- Dzięki nawiasom okrągłym, możemy całe wyrażenie znajdujące się wewnątrz nich przedstawić jako atom.

Przykład:

```
’/(048\-)?/’      ->  kod kraju może wystąpić tylko raz (ale nie musi)
,/([0-9]{3})/’    ->  dokładnie trzy cyfry – traktowane jako atom
```

Atom

Atom – zapisywany w nawiasach ( ) - to najmniejsze wyrażenie we wzorze. Nie rozbijamy go na mniejsze składowe.

- Z atomów zbudowany jest wzór wyrażenia regularnego.
- Atomem może być pojedyncza litera lub cyfra.
- Dzięki nawiasom okrągłym, możemy całe wyrażenie znajdujące się wewnątrz nich przedstawić jako atom.

Przykład:

```
’/(048\-)?/’      ->  kod kraju może wystąpić tylko raz (ale nie musi)
,/([0-9]{3})/’    ->  dokładnie trzy cyfry – traktowane jako atom
```

Funkcja sprawdzająca - preg_match()

Funkcja sprawdzająca - preg_match()

Funkcja preg_match() jest podstawową funkcją PHP dotyczącą wyrażeń regularnych.

Przyjmuje ona minimum dwa argumenty: wzorzec oraz ciąg znaków.

Jeżeli ciąg znaków zostanie pomyślnie dopasowany do wzorca – zwraca wartość logiczną true, w przeciwnym razie false.

```
int preg_match (string $wzorzec, string $ciag_znakow);
```

Funkcja sprawdzająca - preg_match()

Funkcja preg_match() jest podstawową funkcją PHP dotyczącą wyrażeń regularnych.

Przyjmuje ona minimum dwa argumenty: wzorzec oraz ciąg znaków.

Jeżeli ciąg znaków zostanie pomyślnie dopasowany do wzorca – zwraca wartość logiczną true, w przeciwnym razie false.

```
int preg_match (string $wzorzec, string $ciag_znakow);
```

Funkcja sprawdzająca - preg_match()

Przykład:

```php
<?php
    if (preg_match('/abc/', 'abcdabcdabcd'))
    {
        echo "Wzorzec 'abc' pasuje do ciągu znaków 'abcdabcdabcd'";
    }
    else
    {
        echo "Błąd: Wzorzec nie pasuje do ciągu znaków";
    }
?>
```

Funkcja sprawdzająca - preg_match()

Pełna wersja funkcji **preg_match()**

```
preg_match(
    string $pattern,
    string $subject,
    array &$matches = null,
    int $flags = 0,
    int $offset = 0
): int|false
```

Pattern - Wzorzec do wyszukania jako ciąg znaków.

Subject - Ciąg wejściowy.

Matches – tablica zawierającą wszystkie wykryte podłańcuchy

Flags - może być kombinacją następujących flag:

- PREG_OFFSET_CAPTURE - Zwracane zostanie również przesunięcie (w bajtach).
- PREG_UNMATCHED_AS_NULL - niedopasowane podwzorce są zgłaszane jako null; w przeciwnym razie są zgłaszane jako pusty ciąg

Offset - Określa alternatywne miejsce, od którego ma zostać rozpoczęte wyszukiwanie (w bajtach)

Przykład do wykonania

Przykład: Tester wyrażeń regularnych

Funkcja sprawdzająca - preg_match()

Przykład: Tester wyrażeń regularnych

Rozpoczynamy od przygotowania formularza:

```
<form method="post">
      <label for="maska">Maska:</label>
      <textarea class="okno" name="maska" type="text"
               cols="40" rows="5"></textarea><br>
      <label for="wyrazenie">Wyrażenie:</label>
      <textarea class="okno" name="wyrazenie" type="text"
               cols="40" rows="5"></textarea><br>
      <input type="submit" value="Sprawdz">
    </form>
```

Funkcja sprawdzająca - preg_match()

Skrypt php.

Kontener div ułatwi pozycjonowanie elementów

```
<div>
      <?php
      if(isset($_POST['maska']) && isset($_POST['wyrazenie']))
      {
          $maska = $_POST['maska'];
          if(preg_match($maska, $_POST['wyrazenie'],$tablica))
          {
            echo "<h3>Dopasowanie:</h3>";
            foreach($tablica as $klucz=>$wartosc)
            echo "<p>[$klucz]: $wartosc</p>";
          }
          else
          echo "brak dopasowan";
      }
      ?>
    </div>
```

Funkcja sprawdzająca - preg_match()

```
<style type="text/css">
    <!--
    body {
      display: flex;
    }
    form {
      border: 1px solid rgb(67, 88, 159);
      border-radius: 3px;
      padding: 20px;
      margin: 20px 20px;
      display: flex;
      width: 300px;
      flex-direction: column;
    }
    div {
       border: 1px solid rgb(67, 88, 159);
       border-radius: 3px;
      padding: 20px;
      margin: 20px 20px;
      width: 300px;
    }
    -->
```

Na koniec warto dodać arkusz stylu.

Problem z ogonkami - czyli polskie znaki diakrytyczne

Znaki - Unicode character properties

Zbiory [a-z] i [A-Z] nie zawierają polskich znaków.
Dostępne są dodatkowe sekwencje zmiany znaczenia pasujące do ogólnych typów znaków gdy wybrany jest tryb UTF-8. Są to:

\p{xx} - znak z właściwością xx
\P{xx} - znak bez właściwości xx

Przykład:

```
'/\p{L}/u'          ->  dowolna litera
,/\p{Lu}/u'         ->  mała litera,
,/\p{Ll}/u'         ->  duża litera
```

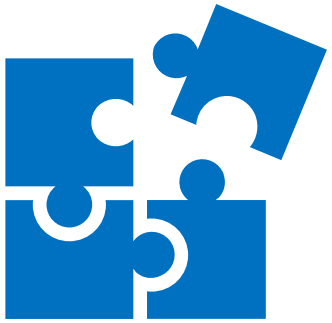
Zwróćmy uwagę na dyrektywę „u” znajdującą się po znaku zamykającym wzorzec. Oznacza ona użycie kodowania UTF8

Znaki

Wybrane znaki
Unicode character properties

Property	Matches	Notes
C	Other	
Cc	Control	
Cf	Format	
L	Letter	Includes : Ll, Lm, Lo, Lt and Lu.
Ll	Lower case letter	
Lm	Modifier letter	
Lo	Other letter	
Lt	Title case letter	
Lu	Upper case letter	
N	Number	
Nd	Decimal number	
Nl	Letter number	
No	Other number	
S	Symbol	
Sc	Currency symbol	
Sk	Modifier symbol	
Sm	Mathematical symbol	
So	Other symbol	
Z	Separator	
Zl	Line separator	
Zp	Paragraph separator	
Zs	Space separator	

Przykład do wykonania



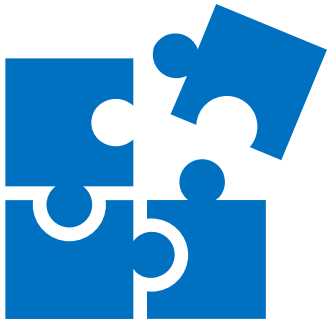
Przykład: Skrypt, który rozbije czas w formie gg:mm na składowe

Funkcja sprawdzająca - preg_match()

```php
<?php
    $czas = '16:34';

    if(preg_match('/^(\d{1,2})\:(\d{1,2})$/D', $czas, $tablica))
    {
        echo '<h3>Dane: "'.$tablica[0].'"</h3>';
        echo '<p>Godzina: '.$tablica[1].'</p>';
        echo '<p>minuta: '.$tablica[2].'</p>';
    }
    else
    {
        echo '<p>Nieprawidłowy format czasu!</p>';
    }
?>
```

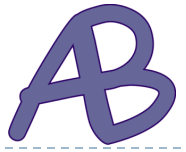
Przykład do wykonania



Przykład: Skrypt, który otrzyma datę w formie ‚10 XII 2022”
i zamieni ją na czas Uniksowy

Funkcja sprawdzająca - preg_match()

```php
<?php
    $data = '1 X 2000';
    if(preg_match('/^(\d{1,2}) (I|II|III|IV|V|VI|VII|VIII|IX|X|XI|XII) (\d{4})$/', $data,
$tablica))
    {
        echo '<h3>Dane: "'.$data.'"</h3>';
        echo '<p>Dzien: '.$tablica[1].'</p>';
        echo '<p>Miesiac: '.$tablica[2].'</p>';
        echo '<p>Rok: '.$tablica[3].'</p>';
        $tablicaKonwertujaca = array('I' => 1, 'II' => 2, 'III' => 3, 'IV' => 4, 'V' => 5, 'VI'
=> 6, 'VII' => 7, 'VIII' => 8, 'IX' => 9, 'X' => 10, 'XI' => 11, 'XII' => 12);
        echo '<p>Unix timestamp: '.mktime(0,0,0,$tablicaKonwertujaca[$tablica[2]], $tablica[1],
$tablica[3]).'</p>';
    }
    else
    {
        echo '<p>Nieprawidłowy format daty</p>';
    }
?>
```



Literatura

W prezentacji użyto przykładów z książki:
- Żygłowicz Jerzy - PHP - Kompendium wiedzy, Helion

- https://www.php.net