COM222 DESENVOLVIMENTO DE SISTEMAS WEB

Aula 16: Expressões Regulares

Conteúdo

- Expressões regulares
  - xxx

O que é regex?

- Método para verificar o “casamento” (matching) de padrões em strings
- Mais comumente usada em PHP: PCRE
  - Perl Compatible Regular Expressions

preg_match()

- Função do PHP que realiza o matching usando regex
- Exemplo:

```
<?php
// cria uma string
$string = 'abcdefghijklmnopqrstuvwxyz0123456789';

echo preg_match("/abc/", $string);
?>
```

  - Retorna 1 se se o padrão for encontrado na string.
  - Retorna 0 caso contrário
  - Retorna FALSE se houver algum erro

Casar com o início de uma string

- O caracter “^” permite checar se uma string começa com um determinado padrão

```
<?php
$string = 'abcdefghijklmnopqrstuvwxyz0123456789';

// Verifica se abc está no começo da string
if(preg_match("/^abc/", $string)){
        echo ‘A string começa com abc';
    }
else {
        echo ‘Nenhum casamento foi encontrado';
    }
?>
```

Case insensitive /i

- preg_match por padrão é case sensitive.
  - Para fazer uma busca por um padrão sem levar em conta a caixa (alta ou baixa), deve-se usar /i

```
<?php
$string = 'abcdefghijklmnopqrstuvwxyz0123456789';

// Verifica se abc está no começo da string
if(preg_match("/^ABC/i", $string)){
        echo ‘A string começa com abc';
    }
else {
        echo ‘Nenhum casamento foi encontrado';
    }
?>
```

  - Retorna 1 para abc, ABC, Abc, aBc, ...

Casar com o fim de uma string

- Para checar se um padrão existe no final de uma string deve-se usar \z

```
<?php
$string = 'abcdefghijklmnopqrstuvwxyz0123456789';

// Verifica se abc está no começo da string
if(preg_match("/89\z/", $string)){
        echo ‘A string termina com 89';
    }
else {
        echo ‘Nenhum casamento foi encontrado';
    }
?>
```

- $ também pode ser usado para fazer o casamento com o final de uma string

Meta caracteres

- Se o padrão buscado contiver um meta caracter, ele deverá aparecer após uma barra invertida

```
<?php
$string = '1+1=2';

// tenta fazer o casamento
if(preg_match("/^1\+1/i", $string))
        {
        echo ‘A string começa com 1+1';
        }
else
        {
        echo 'Nenhum casamento foi encontrado';
        }
?>
```

[ ] : Especificando classes de caracteres

- Classe de caracteres
  - Conjunto de caracteres que se quer fazer match. Ex:
    - [abcdef] ou
    - [a-f]

```
<?php
$string = 'big';

// Procura por um casamento
echo preg_match("/b[aoiu]g/", $string, $matches);
?>
```

  - Retorna 1, pois casa com big. Funcionaria também se $string == bog ou big (mas não beg)

Meta caracteres dentro de classes [ ]

- De modo geral, meta caracteres não funcionam dentro de classes
  - São tratados como um caracter qualquer
    - Ex: [abcdef$]
    - Vai casar com a, b, c, d, e, f, $
- Exceção
  - ^ no início da classe
  - Permite que se encontre padrões *fora* do intervalo especificado pela classe

^ no início de uma classe


```
<?php
$string = 'abcefghijklmnopqrstuvwxyz0123456789';

// procura por um casamento
preg_match("/[^b]/", $string, $matches);

// Faz loop imprimindo os casamentos
foreach($matches as $key=>$value) {
        echo $key.' -> '.$value;
        }
?>
```

- Resultado
  - 0 -> a

^ no início de uma classe

- Agora usando preg_match_all()

```
<?php
$string = 'abcefghijklmnopqrstuvwxyz0123456789';

// tenta fazer o casamento com todos os caracteres fora do padrão
preg_match_all("/[^b]/", $string, $matches);

//  Faz loop imprimindo os casamentos
foreach($matches[0] as $value){
        echo $value;
        }
?>
```

- Resultado
  - acefghijklmnopqrstuvwxyz0123456789

^ no início de uma classe

- Filtrando números de uma string

```
<?php
$string = 'abcefghijklmnopqrstuvwxyz0123456789';

// Casa com qualquer caracter que não seja um número (de 0 a9)
preg_match_all("/[^0-9]/", $string, $matches);

//  Faz loop imprimindo os casamentos
foreach($matches[0] as $value){
        echo $value;
        }
?>
```

- Resultado
  - abcefghijklmnopqrstuvwxyz

O meta caracter \ barra invertida

- Usado para “escapar” outros meta caracteres, incluindo ele próprio

```
<?php
$string = 'This is a [templateVar]';

//  procura por um casamento
preg_match_all("/[\[\]]/", $string, $matches);

//  Faz loop imprimindo os casamentos
foreach($matches[0] as $value){
        echo $value;
        }
?>
```

- Resultado: []

O meta caracter . ponto

- O . casa com qualquer caracter, com exceção de quebra de linha: \r ou \n

```
<?php
$string = 'sex at noon taxes';

//  procura por um casamento
echo preg_match("/s.x/", $string, $matches);

?>
```

- Resultado: 1
- Também casaria com sax, six, sox, sux e s x

Como casar com quebra de linha: \s


```
<?php
$string = 'sex'."\n".'at'."\n".'noon'."\n".'taxes'."\n";

// imprime a string
echo nl2br($string);

//  procura por um casamento
echo preg_match_all("/\s/", $string, $matches);

?>
```

- Resultado:

  sex
  at
  noon
  taxes
  4

O meta caracter * asterisco

- O * casa com zero ou mais ocorrências do caracter que o precede

```
<?php
$string = 'php';

// procura por um casamento
echo preg_match("/ph*p/", $string, $matches);

?>
```

  - Resultado: 1
  - Também casaria com pp phhp phhhhp ....

O meta caracter + mais

- O + casa com uma ou mais ocorrências do caracter que o precede

```
<?php
$string = 'pp';

// procura por um casamento
echo preg_match("/ph+p/", $string, $matches);

?>
```

- Resultado: 0

O meta caracter + mais

- O + casa com uma ou mais ocorrências do caracter que o precede

```
<?php
$string = 'pp';

// procura por um casamento
echo preg_match("/ph+p/", $string, $matches);

?>
```

- Resultado: 0

O meta caracter ?

- O ? casa com zero ou uma ocorrência do caracter que o precede

```
<?php
$string = '12345678';

// procura por um casamento
echo preg_match("/1234-?5678/", $string, $matches);

?>
```

- Resultado: 0
- O exemplo acima permite validar um número de telefone de 8 dígitos com ou sem hífem.

Exercício 1

- Como validar um número de telefone com 8 ou 9 dígitos, com ou sem hífem?