EXPRESSÕES REGULARES
(ER, Ereg ou RegEx)

- **Uma expressão regular é essencialmente um padrão. É com base nesse padrão que a expressão em questão será avaliada.**

- **Uma expressão regular, na Informática, define um padrão a ser usado para procurar ou substituir palavras ou grupos de palavras. É um meio preciso de se fazer buscas de determinadas porções de texto.**

- **São um método rápido e simples de manipulação e combinação avançada de strings.**

Caracter ^

Este é o primeiro operador que vamos aprender. O ^ indica que a expressão deve iniciar com a string dada.

```
^flu
```

Esta sintaxe irá definir que a string dada deverá começar com 'flu'. No caso, "fluzão é a máquina" seria uma string concordante. Porém, se você tentar "Após a goleada o flu terá descanso", não obterá sucesso.

Caracter $

Como existe o ^ para definir o começo, temos $ para definir o final. Com $ defini-se a string que deverá cocordar com o final.
flu$

Esta sintaxe irá definir que a string dada deverá terminar com 'flu'. Tentando validar a string "E o campeão é o flu" obteremos sucesso, o que não acontecerá com a string "O flu foi o campeão".

Outros Caracteres

**** - É utilizado para não interferir na função utilizada.
Ex: Se você por um "(" para comparar, ela irá interferir na função
Solução: colocar um \(, assim também com as aspas \".

***** - Identifica qualquer seqüência de caracteres.

? - Identifica um caractere qualquer.

{x} - Traz "x" caracteres de uma seqüência (x deve ser número natural).

{x, y} - Traz entre "x" e "y" caracteres de uma seqüência (x e y devem ser números naturaisl).

. - Identifica qualquer caractere.

x|y - O "|" identifica uma condição, no caso se não for o caractere "x" irá ser o "y".

Outros Caracteres

\d - Marca uma seqüência de caracteres números.

\D - Marca uma seqüência de não-números.

\w - Marca uma seqüência de caracteres alfabéticos.

\W - Marca uma seqüência de caracteres não-alfabéticos.

\s - Identifica caracteres “em branco” tipo o barra de espaço.

Caracteres em Branco e Caracteres de Escape

Assim como na sintaxe PHP, em expressões regulares também tem que utilizar de caracteres de escape, como:

\n - Nova linha
\f - Avanço de página
\r - Retorno de carro
\. - Qualquer caractere
\\ - Uma barra invertida literal
\- - Um hífen literal

Classes de Caracteres

Classes de caracteres são uma espécie de grupo que contêm todas as ocorrências que deverão (ou não deverão) constar em uma expressão

Ex: Para permitir apenas letras vogais, usa-se:
[AaEeIiOoUu]

Então com este código teríamos um padrão que só permitirá que apenas vogais estejam na string dada.

Intervalo de caracteres
[a-z] - Qualquer letra minúscula
[A-Z] - Qualquer letra maiúscula
[a-zA-Z] - Qualquer letra maiúscula ou minúscula
[0-9] - Qualquer número
[0-9\.\-] - Qualquer número, ponto ou sinal de subtração

Isto serve apenas para combinação de UM caractere.

Classes de Caracteres

^[a-z][0-9]$

A expressão de apenas dois caracteres em que o primeiro tem necessariamente que ser uma letra minúscula e o segundo ser um número.

ATENÇÃO: operador ^! fora de uma classe de caracteres, ele serve para demonstrar o que deve haver no início, porém dentro, ele serve para mostrar o que não deve ocorrer.

^[**^**0-9][0-9]$

A expressão deve começar com um caractere que não seja um número e seja imediatamente seguido por um caractere que será um número.

Classes de Caracteres Pré-definidas

As classes de caracteres pré-definidas que já vêm junto com o interpretador de ER que você estiver utilizando (apresentando apenas o método POSIX).

[[:alpha:]] // Qualquer letra (alfabético)
[[:digit:]] // Qualquer número (dígito)
[[:alnum:]] // Qualquer letra ou número (alfanumérico)
[[:space:]] // Qualquer caractere de espaço
[[:upper:]] // Qualquer letra maíuscula
[[:lower:]] // Qualquer letra minúscula
[[:punct:]] // Qualquer caractere de pontuação
[[:xdigit:]] // Qualquer dígito hexadecimal (Equivalente a: [0-9a-fA-F])

Ocorrências Múltiplas

Agora vamos pra parte em que realmente começa a fazer sentido usar-se expressões regulares.

^[[:alpha:]]{3}$ // Qualquer palavra de três letras
^a{4}$ // Só fecha com a expressão 'aaaa'
^a{2,4}$ // Fecha com 'aa', 'aaa' e 'aaaa'
^a{2,}$ // Fecha com 'aa', 'aaa', 'aaaa', 'aaaaa', ...

Outras formas de representar a repetição de caracteres

? - Uma ocorrência ou nenhuma (Equivale a {0,1})
* - Nenhuma ocorrência, uma ocorrência, duas ocorrências,... (Equivale a {0,})
+ - Uma ou mais ocorrências (Equivale a {1,})

Alternação e Parênteses

Lógico OR (|)

f|c é equivalente a [fc]

Para caracteres simples é com certeza mais conveniente utilizar as classes. A alternação é basicamente útil para utilizar alternância de palavras.
Arroz|Feijão|Angu

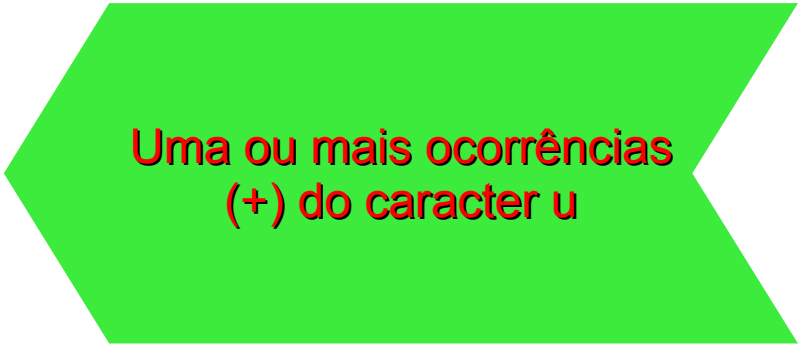
Esta expressão só irá retornar verdadeiro se a string dada for "Arroz" OU "Feijão" OU "Angu".

Alternação e Parênteses

Parênteses ()

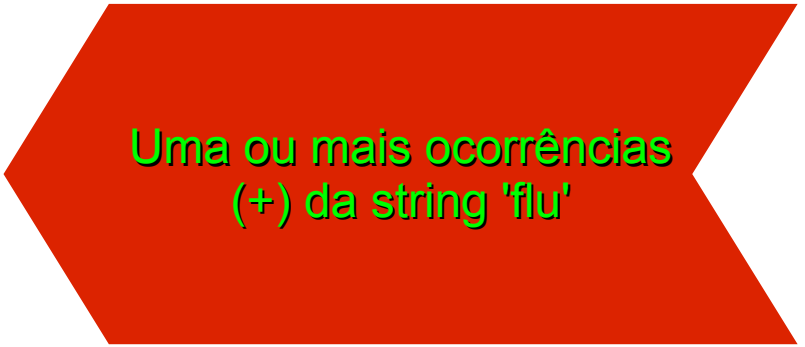
ex: Flu+

Somente fecharia com:
Flu
Fluu
Fluuu
Fluuuu...



Porém com os parênteses, temos um jeito mais interessante de interar com palavras repetidas.
(flu)+

Irá fechar com:
flu
fluflu
flufluflu



Alternação e Parênteses

(ci|experi|decad)encia

Irá retornar verdadeiro com:
ciencia
experiencia
decadencia

Arroz|Feijão$ - Retorna verdadeiro com 'Arroz' em qualquer parte da string ou com 'Feijão' no final

(Arroz|Feijão)$ - Retorna verdadeiro tanto com "Arroz" quanto "Feijão", porém, no final da string

([ab])([xy]) - Retorna verdadeiro com 'ax', 'ay', 'bx' e 'by' em qualquer parte da string

Aplicando Expressões Regulares em PHP

EREG

bool ereg (string expressao, string variavel [, array registros])

Verifica se a variavel casa com a expressão regular definida em expressão em um modo sensível a distinção de caracteres (case sensitive).

Se existirem parenteses de substrings na expressão e for passado o terceiro parâmetro (registros) para a função, a execução guardará os elementos resultantes na matriz registros.

$registros[1] irá conter a substring indicada pelo primeiro parenteses da esquerda;
$registros[2] contém a segunda substring, e assim por diante.
$registros[0] conterá uma cópia completa da variavel casada.

Ex: $data = “2008-03-13”;
ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $data, $registros)

$registros[0]= 2008-03-13
$registros[1]= 2008
$registros[2]= 03
$registros[3]=13

EREG

Nota: Nas versões superiores ao PHP 4.1.0 (inclusive) se você colocar exatamente dez elementos em $registros, ou até mesmo mais elementos, a expressão será executada.

A quantidade não causará efeitos na função ereg() que tem habilidade de suportar muitas substrings. Se a expressão não casar, $registros não será alterada por ereg().

EREGI

bool eregi (string expressao, string variavel [, array registros])

Essa função é idêntica a função ereg() com exceção de não fazer distinções alfabéticas entre caracteres (case insensitive) na hora de casar resultados.

```
Ex:  $variavel = "Zorro !";
      if (eregi("z", $variavel))
     {
      echo "A '$variavel' contém a letra 'z' ou 'Z'!";
      }
```

EREG_REPLACE

string ereg_replace (string expressao, string substituicao, string variavel)

Essa função busca em variavel resultados para a expressão, substituindo se casar pelo texto em substituição.

A variavel modificada será retornada (poderá ocorrer da string original ser retornada caso não aconteça nenhuma substituição).

Por exemplo, o pedaço de código seguinte imprimirá "Esse foi um teste"

Ex: $string = "Esse e um teste
";
    echo ereg_replace (" e", " foi", $string);

EREG_REPLACE

Atenção: Usar uma variável integer no parâmetro substituição, pois o resultado pode não ser exatamente o esperado.

Ex: $num = 5;
```
$string = "Essa frase tem cinco palavras<br>";
$string = ereg_replace('cinco', $num, $string);
echo $string;   // Resultado: 'Essa frase tem   palavras.'
```

Isso acontece porque a função ereg_replace() interpreta o valor ordinal do número.

Ex:$num = '5';
```
$string = "Essa frase tem cinco palavras <br>";
$string = ereg_replace('cinco', $num, $string);
echo $string;   // Resultado: 'Essa frase tem 5 palavras.'
```

EREG_REPLACE

Ex: Substitui URLs por links

```
$text = ereg_replace("[[:alpha:]]+://[^<>[:space:]]+[[:alnum:]/]",
"<a href=\"\\0\">\\0</a>", $text);
```

EREG_REPLACE

Ex: Substitui URLs por links

```
$text = ereg_replace("[[:alpha:]]+://[^<>[:space:]]+[[:alnum:]/]",
"<a href=\"\\0\">\\0</a>", $text);
```

EREGI_REPLACE

string eregi_replace (string expressão, string substituição, string variavel)

Essa função é idêntica a ereg_replace() com exceção de não fazer distinções alfabéticas entre caracteres (case insensitive) na hora de casar resultados.

STR_REPLACE

mixed str_replace (mixed pesquisa, mixed substitui, mixed assunto [, int &count])

Esta função retorna uma string ou um array com todas as ocorrências de pesquisa em assunto substituidas com a o valor dado para substitui.

Ex: $string = " o @ é usado em endereços de e-mail como rafa@processware.com.br";

```
echo str_replace ("@", " ARROBA ", $string);
```

Substituindo valores na string usando Arrays

Ex: $frase = "você torce para o Santos de Pelé e Robinho." ;

```
$antigo = array("Santos", "Pelé", "Robinho");
$novo = array("Fluzão", "Washington", "Dodo");
$novafrase = str_replace($antigo, $novo, $frase);
echo $novafrase;
```

STR_REPLACE

ATENÇÃO: Uso do parâmetro count está disponível no PHP 5.0.0

Ex: $str = str_replace("r", "", "O rato roubou o remédio", $count);
echo $count;

EXPRESSÕES REGULARES
“Mais comuns...”

Validação de E-Mail

```
$string = "email@email.com";
$pattern = "/(\w{3,15})@{1}(\w{2,15})\.com(\.)?(\w{2})?/";
if(preg_match($pattern,$string,$veto))
{
echo $veto[0];
}
else
{
echo "E-mail inválido";
}
```

PREG_MATCH

preg_match verifica se uma ER casa (retorna true) ou não casa (retorna false)

Validação de E-Mail

```
$email = "email@email.com";
$express = "/(\w{3,15})@{1}(\w{2,15})\.com(\.)?(\w{2})?/";
if(preg_match($express,$email,$saida))
{
echo $saida[0];
}
else
{
echo "E-mail inválido";
}
```

Entendendo a expressão:

/(\w{3,15}) - Verifica se existe um conjunto de letras entre 3 e 15 caracteres.
@{1} – Avalia se existe um arroba.
(\w{2,15}) - Verifica se existe um conjunto entre 2 e 15 letras.
\.com - Avalia se existe um .com
(\.)? –Condição que indica se existe um . após o com
(\w{2})?/ - Avalia se existe 2 letras.

Números Telefônicos

Considerando o formato: xxx – xxxx-xxxx (com DDD)
Máscara: (2 ou 3 numeros) – (4 a 7 numeros) – (4 numeros)

```
$string = "021-2567-4324";
if (preg_match('/^([0-9]{2,3})([-]?[0-9]{4,7})([- ]?[0-9]{4,4})?$/', $string)) {
echo "telefone valido.";
}
else
{
echo "telefone invalido.";
}
```

Considerando o formato: xxxx-xxxx (sem DDD)
Máscara:(4 a 7 numeros) – (4 numeros)

```
$string = “2567-4324";
if (preg_match('/^[0-9]{4,7}([- ]?[0-9]{4,4})?$/', $string)) {
echo "telefone 2 valido.";
}
else
{
echo "telefone 2 invalido.";
}
```

Validando uma URL

```
$url = "xxxxxxxxxxhttp://www.yart.com.brmnopqrstuvxz";
$validador = "/([http:\/\/]{7})?w{3}\.{1}?[a-z]{3,15}\.[com]{3}(\.)?(\w{2})?/";
preg_match($validador,$url,$vet);
echo $vet[0];
```

Entendendo a expressão:

([http:\/\/]{7})? – caso exista um http://

w{3}\.{1}? – caso exista 3 w (www) e após um ponto final

[a-z]{3,15} – depois exista uma seqüência de a à z entre 3 e 15 letras

\.[com]{3} – caso exista um .com

(\.)?(\w{2})? - E depois de tudo avalia caso exista ou não um . mais dois caracteres (.br ou .uk)

Validando um endereço IP

```
$ip = "120.0.0.100";
$teste = "/^(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9]
[0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)
$/";
if(preg_match($teste,$ip,$res))
{
echo "valida o endereco   ". $res[0];
}
else
{
echo "nao valida o endereco   ". $ip;
}
```

Entendendo a expressão:

/^(25[0-5] – Inicia com 25x, onde x = {0...5}
|- ou
2[0-4][0-9] - Inicia com 2 xy, onde x = {0...4} e y = {0...9}
|- ou
[01]?[0-9][0-9]?) - Inicia com 0 ou 1, seguido de {0...9} e {0...9}

o mesmo vale para o restante da expressão

Validando um endereço IP

Observe que pode-se fazer o mesmo com funções, porém expressões regulares são mais práticas

```
function ipValida($ip)
{
    if(!eregi("^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$", $ip))
    return false;

$tmp = explode(".", $ip);
    foreach($tmp as $sub){
    $sub = $sub * 1;
        if($sub<0 || $sub>256) return false;
    }
    return true;
    }
```

Todas as expressões regulares apresentadas neste resumo possuem erros de lógica de validação.

Ex: na expressão de validação de e-mail, só testamos se existe o .com após o domínio, assim .comxxxx retorna como válido e não é
Um outro erro é de que existem domínios como .eti, .inf, .br ,e muitos outros que são válidos e a expressão retorna como falso.
Na validação do nome, não é validado números e (ex: email123@email.com) isso é possível
Existem erros de lógica em todas as expressões !

O objetivo deste documento é apresentar a sintaxe básica de RegEx.

Arrume os erros das expressões apresentadas, com um exercícios prático sobre o assunto.