PHP – PARTE 3

DIOGO CEZAR TEIXEIRA BATISTA
http://inf.cp.utfpr.edu.br/diogo
diogo@diogocezar.com

COMANDO EXIT

- comando **exit()** encerra a execução do script php;
- utilizado quando não se deseja continuar exibindo a página.

```
<?php
$permissao = false;
if(!$permissao){
   echo "Você não pode acessar essa
página!";
   exit();
}
?>
```

UTILIZANDO A FUNÇÃO LIST

- a função list() quebra um array em vários valores:
  - captura o valor do array e os atribui para a lista de variável passada como parâmetro;

```
<?php
$info = array('Café', 'marrom', 'cafeína');

// Listando todas as variáveis
list($bebida, $cor, $substancia) = $info;
echo "$bebida é $cor e $substancia o faz especial.";
?>
```

OUTRAS FUNÇÕES DE CLASSIFICAÇÃO

- asort() - classifica um array associativo crescentermente, ordenando pelos valores:

```
<?php
$frutas = array("d" => "limão",  "a" => "laranja",
                "b" => "banana", "c" => "maçã");
asort($frutas);
foreach($frutas as $chave => $valor ){
    echo "$chave = $valor\n";
}
?>
```

O exemplo acima irá imprimir:

```
b = banana
a = laranja
d = limão
c = maçã
```

OUTRAS FUNÇÕES DE CLASSIFICAÇÃO

- ksort() - classifica um array associativo crescentemente, ordenando pelas chaves:

```
<?php
$frutas = array("d" => "limão",  "a" => "laranja",
                "b" => "banana", "c" => "maçã");
ksort($frutas);
foreach($frutas as $chave => $valor ){
    echo "$chave = $valor\n";
}
?>
```

O exemplo acima irá imprimir:

```
a = laranja
b = banana
c = maçã
d = limão
```

OUTRAS FUNÇÕES DE CLASSIFICAÇÃO

- rsort() - classifica um array numérico decrescentemente:

```
<?php
$frutas = array ("limão", "laranja", "banana", "maçã");
rsort($frutas);
foreach($frutas as $chave => $valor ){
    echo "$chave = $valor\n";
}
?>
```

O exemplo acima irá imprimir:

```
0 = maçã
1 = limão
2 = laranja
3 = banana
```

Nota: Esta função define novas chaves para os elementos em *array*.
Ela irá remover qualquer chave que você tenha definido

OUTRAS FUNÇÕES DE CLASSIFICAÇÃO

- arsort() - classifica um array associativo decrescentemente, ordenando pelos valores:

```
<?php
$frutas = array("d" => "limão",  "a" => "laranja",
                "b" => "banana", "c" => "maçã");
asort($frutas);
foreach($frutas as $chave => $valor ){
    echo "$chave = $valor\n";
}
?>
```

O exemplo acima irá imprimir:

```
c = maçã
d = limão
a = laranja
b = banana
```

OUTRAS FUNÇÕES DE CLASSIFICAÇÃO

- krsort() - classifica um array associativo decrescentemente, ordenando pelas chaves:

```
<?php
$frutas = array("d" => "limão",  "a" => "laranja",
                "b" => "banana", "c" => "maçã");
krsort($frutas);
foreach($frutas as $chave => $valor ){
    echo "$chave = $valor\n";
}
?>
```

O exemplo acima irá imprimir:

```
d = limão
c = maçã
b = banana
a = laranja
```

FUNÇÕES PARA REORDENAÇÃO

- shuffle($nomeDoArray) – ordena aleatoriamente os elementos do array;

```
<?php
$frutas = array('d' => 'limão', 'a' => 'laranja',
                'b' => 'banana','c' => 'maça');
shuffle($frutas);
foreach ($frutas as $chave => $valor) {
        echo $chave." = ".$valor."<br>";
}
?>
```

O exemplo acima irá imprimir:

```
0 = laranja
1 = banana
2 = limão
3 = maça
```

Nota: Esta função define novas chaves para os elementos em *array*.
Ela irá remover qualquer chave que você tenha definido

FUNÇÕES PARA REORDENAÇÃO

- array_reverse($nomeDoArray) - cria um novo array como o mesmo conteúdo do original só que na ordem inversa.

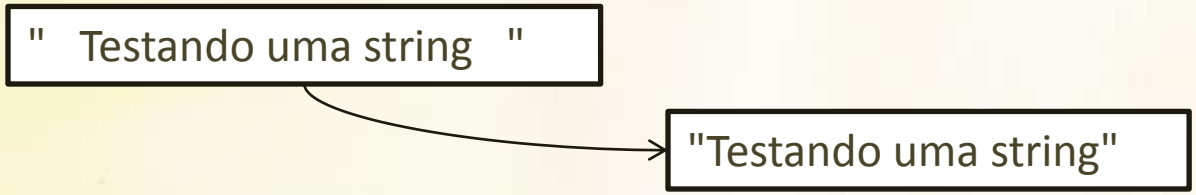
```
<?php
$frutas = array('d' => 'limao', 'a' => 'laranja',
                'b' => 'banana', 'c' => 'maça');
$frutas_reverse = array_reverse($frutas);
foreach ($frutas_reverse as $chave => $valor) {
        echo $chave." = ".$valor."<br>";
}
?>
```

O exemplo acima irá imprimir:

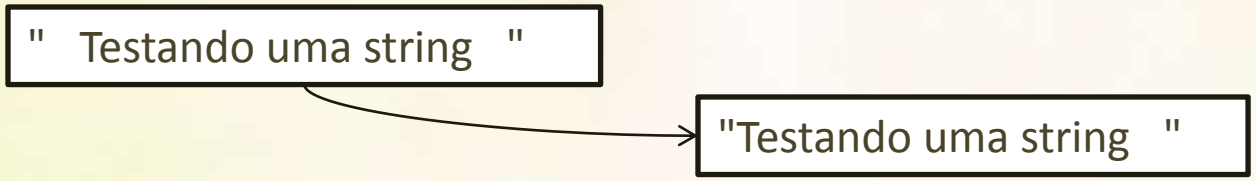
```
c = maça
b = banana
a = laranja
d = limão
```

FORMATANDO STRINGS

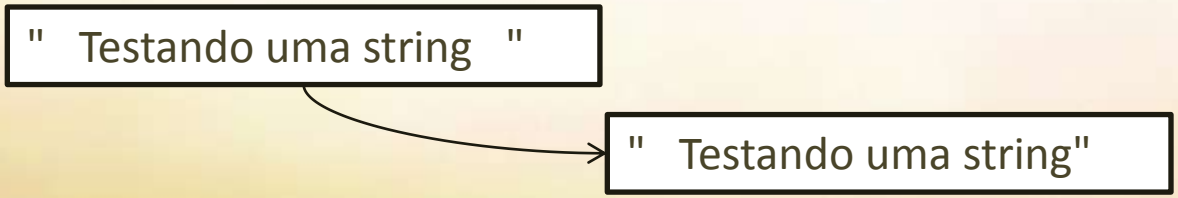
- trim() - elimina os espaços em branco do início e do final da string, retornando a string resultante:



- ltrim() - elimina somente os espaços em branco do início:



- chop() - elimina somente os espaços em branco do final:



FORMATANDO STRINGS

- nl2br() - substitui as novas linhas da string ('/n') pela tag
 do html;
  - Útil ao exibir textos vindos de <textarea> pois retornam o texto com as quebras de linhas;

```
<?php
$texto = "Tetando \n uma nova linha";
echo nl2br($texto);
?>
```

O exemplo acima irá imprimir:

```
Tetando
uma nova linha
```

ALTERANDO A CAIXA DE UMA STRING

- strtoupper() - coloca a string toda em letras maiúsculas;
- strtolower() - coloca a string toda em letras minúsculas;
- ucfirst() - coloca o primeiro caractere da string em letra maiúscula;
- ucwords() - coloca o primeiro caractere de cada palavra em letra maiúscula.

ALTERANDO A CAIXA DE UMA STRING

```
<?php
$str = "testando uma string";
echo strtoupper($str);
echo "<br>";
echo strtolower($str);
echo "<br>";
echo ucfirst($str);
echo "<br>";
echo ucwords($str);
?>
```

O exemplo acima irá imprimir:

```
TESTANDO UMA STRING
testando uma string
Testando uma string
Testando Uma String
```

Caracteres especiais

- em php quando queremos construir uma string com um caractere especial, devemos inserir antes do caractere uma '\' para que o interpretador entenda aquilo como uma string, por exemplo:

```
<?php
$str = "testando uma \"string\"";
echo $str;
?>
```

O exemplo acima irá imprimir:

```
testando uma "string"
```

FORMATANDO STRINGS PARA ARMAZENAMENTO

- AddSlashes() - Adiciona automaticamente uma barra invertida (\) antes de caracteres especiais;
- StripSlashes() - Remove as barras invertidas (\) localizadas antes de caracteres especiais;

```
<?php
$str = "Seu nome é O'reilly?";
echo addslashes($str);
echo "<br>";
echo stripslashes($str);
?>
```

O exemplo acima irá imprimir:

```
Seu nome é O\'reilly?
Seu nome é O'reilly?
```

REUTILIZAÇÃO DE CÓDIGO

- usamos require() ou include() para inserir um outro arquivo no arquivo corrente;
  - Se for um outro arquivo php, pode-se utilizar todos os recuros oferecidos pelo código;
- a diferença entre require() ou include() é que ao utilizar require() caso não se encontre o arquivo desejado o script termina a sua execução (Fatal Error);
- require() só adiciona um trecho de código caso ele ainda não tenha sido adicionado;

REUTILIZAÇÃO DE CÓDIGO

include_funcao.php

```
<?php
function soma($a, $b){
	return ($a + $b);
}
?>
```

include_principal.php

```
<?php
include("include_funcao.php");
echo soma(10, 15);
?>
```

O exemplo acima irá imprimir:

```
25
```

UTILIZANDO FUNÇÕES

- as vantagens de se utilizar funções são:
  - maior legibilidade ao código;
  - possibilidade de reutilização;
  - podem receber valores (parâmetros) – quando for necessário que a funcionalidade varie de acordo com uma situação particular;
  - podem retornar valores – como resposta ao processamento executado;
  - podem ser agrupadas em um único arquivo e utilizada em todas páginas por um include.

CRIANDO SUAS PRÓPRIAS FUNÇÕES

- regras para nomes:
  - distinguem maiúsculas de minúsculas;
  - não pode ter o mesmo nome que uma função pré-existente;
  - só pode conter letras, dígitos e sublinhados;
  - não pode iniciar com um dígito.
- uma função em php pode ter um parâmetro opcional, definido em sua declaração:

```
function divide($a, $b, $verificaDivZero = true){
```

Nota: Não é necessário passar um terceiro valor para a função divide;

CRIANDO SUAS PRÓPRIAS FUNÇÕES

- passagem de parâmetro por valor:
  - uma cópia da variável passada é criada;
  - não há qualquer alteração com a variável original;
- passagem de parâmetro por referência:
  - a variável original (exterior à função) é alterada;
  - utilize um e comercial (&) antes do nome do parâmetro na declaração da função.

```
function soma(&$a, &$b){ ... }
```

MANIPULANDO DATAS

- strtotime() - analisa qualquer descrição em texto em inglês de data e hora em timestamp Unix;
  - com essa função é possível capturar um timestamp de datas passadas ou futuras;
  - imagine que hoje é dia 27/01/2009 e você precisa saber que dia vai ser daqui a 9 dias?
    - criar função que conta os dias do mês?
    - ano bissexto?
    - mês com 31 dias?
  - com esta função basta "somar" "+9 days" a data atual, e a data final é gerada automaticamente.

MANIPULANDO DATAS

```
<?
$dataAtual = mktime(0,0,0,1,27,2009);
$dataNova  = strtotime("+9 days", $dataAtual);
echo date("d/m/Y", $dataNova); //05/02/2009
?>
```

```
<?php
echo strtotime("now");
echo strtotime("10 September 2000");
echo strtotime("+1 day")";
echo strtotime("+1 week");
echo strtotime("+1 week 2 days 4 hours 2 seconds");
echo strtotime("next Thursday");
echo strtotime("last Monday");
?>
```

Nota: A string para analisar, de acordo com a sintaxe GNU de Formato de Entrada de Data.

EXTRAINDO DADOS DE FORMULÁRIOS

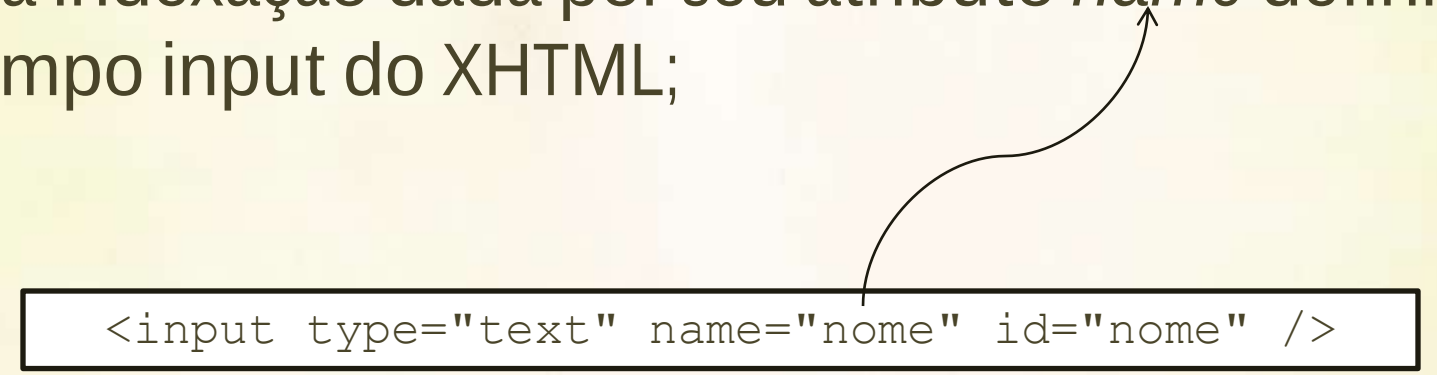
- vimos que é possível trafegar informações pelas páginas web de 2 formas:
  - método GET (Variáveis anexas na URL da página)
  - método POST (Variáveis escritas no cabeçalho do protocolo HTTP);
- vamos ver como capturar essas variáveis em PHP:

MÉTODO POST

- para capturar as variáveis enviadas pelo método POST, a linguagem php define um array global:

$_POST[];

- as variáveis enviadas estarão nesse array global com sua indexação dada por seu atributo *name* definido no campo input do XHTML;



```
<input type="text" name="nome" id="nome" />
```

EXEMPLO

```
<html>
<head>
<title>Método POST</title>
</head>
<body>
<?php
   if(!empty($_POST['nome'])){
      echo "<h1>Seu nome é: ".$_POST['nome']."</h1>";
   }
   else{
   ?>
      <form method="post" action="metodo_post.php">
      <label>Nome:
      <input type="text" name="nome" id="nome" />
      </label>
      <input type="submit" name="enviar" id="enviar" value="Enviar" />
      </form>
<?php
   }
?>
</body>
</html>
```

variável global $_POST recuperando a variável 'nome'

Para onde o formulário será enviado

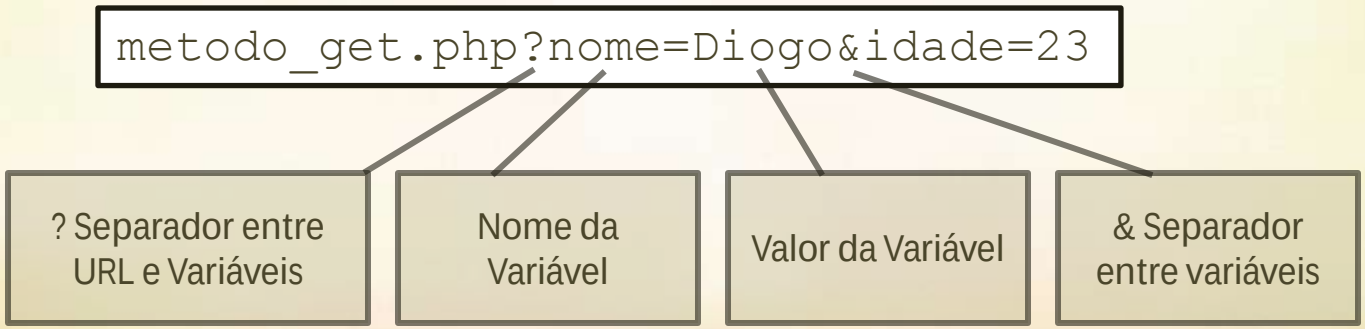
Nome da variável que será recuperada

Método GET

- Para capturar as variáveis enviadas pelo método GET, a linguagem php define um array global:

$_GET[];

- As variáveis capturadas pelo método GET são passadas por um formulário ou por um link diretamente na URL do arquivo, por exemplo:



EXEMPLO

```
<?php
        $nome =  $_GET['nome'];
        $idade = $_GET['idade'];

        if(empty($nome) || empty($idade)){
                echo "Nao foram encontradas todas as variaveis
de GET, utilize: metodo_get.php?nome=Diogo&idade=23";

        }
        else{
                echo "Ola, $nome. Voce tem $idade anos.";
        }
?>
```

Variável global $_GET recuperando a variável 'nome' e 'idade'