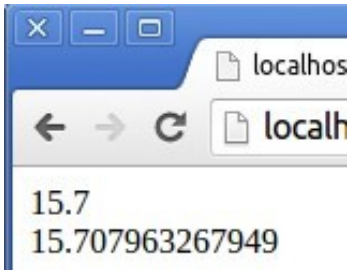
INSTITUTO FEDERAL
SANTA CATARINA
Câmpus Canoinhas

Funções

Funções

Funções são trechos de código que podem realizar qualquer tipo de tarefa, como, por exemplo, calcular a área do círculo. Para definir uma função, é utilizada a palavra reservada **function**.

```php
<?php
function calcularAreaCirculo($raio) {
    return 3.14 * $raio;
}
function calcularAreaCirculo2($raio) {
    return M_PI * $raio;
}
$raio = 5;
echo calcularAreaCirculo($raio);
echo '<br />';
echo calcularAreaCirculo2($raio);
?>
```



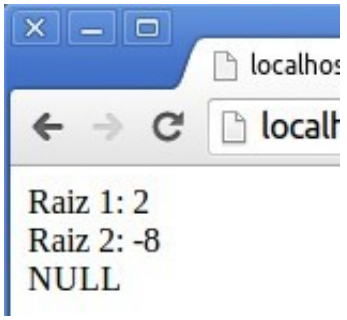
Funções

Podemos chamar outras funções de dentro da nossa função e até fazer funções recursivas. O exemplo a seguir calcula a fórmula de Bhaskara e imprime na tela as duas raízes. Nem sempre a função precisa ter um retorno em PHP.

- sqrt($variavel) calcula a raiz quadrada pow($base, $expoente) calcula elevado
- var_dump($variavel) imprime na tela o valor e o tipo da variável
-

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```php
<?php
function bhaskara($a, $b, $c) {
    $delta = sqrt( pow($b, 2) - (4 * $a * $c) );
    $raiz1 = (-$b + $delta) / 2 * $a;
    $raiz2 = (-$b - $delta) / 2 * $a;
    echo "Raiz 1: $raiz1 <br />Raiz 2: $raiz2";
}
$retorno = bhaskara(2, 3, -2);
echo '<br />';
var_dump($retorno);
?>
```



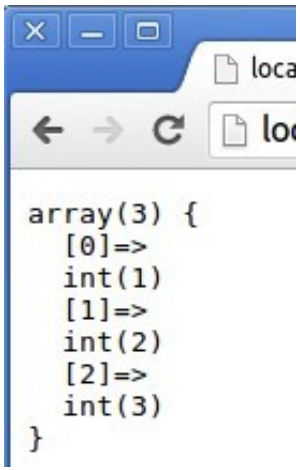
Funções

A passagem de parâmetros nas funções é feita por **valor**. Isso significa que se alterar o valor de uma variável dentro da função, não reflete no valor da variável fora da função.

No caso de **objetos**, a passagem é feita por referência.

É possível forçar a passagem por referência colocando um & antes do parâmetro na declaração da função.

```php
<?php
function alterarValor($vetor) {
    $vetor[0] = 45;
}
$vetor = array(1,2,3);
alterarValor($vetor);
echo '<pre>';
var_dump($vetor);
echo '</pre>';
?>
```

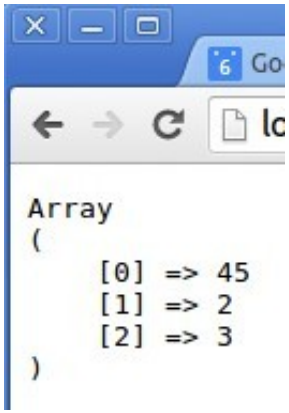


Funções

Nós utilizamos nos exemplos anteriores a função **var_dump**. Ela mostra o valor e o tipo de uma variável. Muitas vezes nós não precisamos saber o tipo. Existe uma função muito parecida que mostra somente os valores: **print_r**.

Perceba que no exemplo de agora, foi colocado um & antes do parâmetro. Isto significa que estamos passando o parâmetro por referência.

```php
<?php
function alterarValorDeVerdade(&$vetor) {
    $vetor[0] = 45;
}
$vetor = array(1,2,3);
alterarValorDeVerdade($vetor);
echo '<pre>';
print_r($vetor);
echo '</pre>';
?>
```

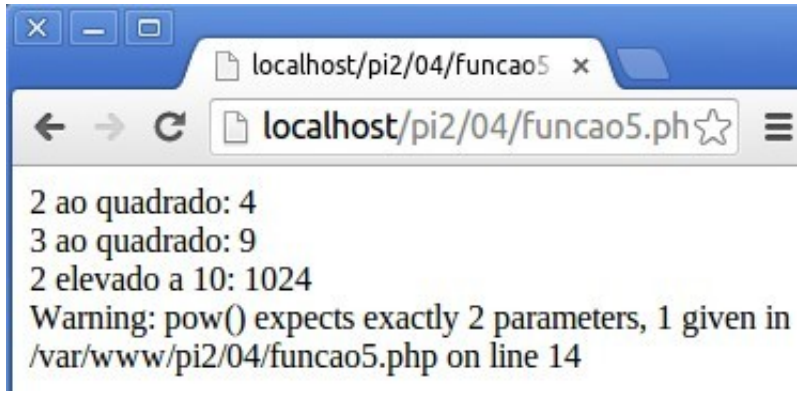


Funções

Um recurso muito útil no PHP é que é possível definir valores padrão para os argumentos da função. Caso a função seja chamada sem passar um parâmetro, o valor padrão é utilizado. É possível que uma função tenha todos ou nenhum parâmetro padrão. Os parâmetros padrão devem ser os últimos declarados.

Se não for definido um valor padrão e não for passado um parâmetro, então ocorrerá um erro.

```php
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
<?php
function elevado($base, $expoente = 2) {
    return pow($base, $expoente);
}
echo '2 ao quadrado: ' . elevado(2, 2). '<br />';
echo '3 ao quadrado: ' . elevado(3). '<br />';
echo '2 elevado a 10: ' . elevado(2,10). '<br />';
pow(2);
?>
</body>
</html>
```



Funções Nativas

O PHP já vem com centenas (ou talvez milhares) de funções nativas. Isso significa que não é necessário instalar extensões adicionais para a maioria das tarefas comuns.

Além das funções nativas, também existem extensões que podem ser instaladas para utilizar outros tipos de recursos, como, por exemplo, manipulação de arquivos utilizando funções de baixo nível.

Funções Nativas

A seguir estão alguns tipos de funções do PHP:

Cache	Depuração	Áudio
Autenticação	CLI	Compressão
Cartão de crédito	Criptografia	Banco de dados
Data e hora	Sistema de arquivos	Imagem
Email	Matemática	Processos
JSON	Sessão	Processamento de texto
Reflexão	Serviços web	XML

Funções Nativas

array

Cria um array.

bool array_key_exists($chave, $array) Verifica se a chave existe.

array array_keys($array)

Retorna um array com as chaves como valores.

number array_sum($array)

Soma os elementos. array asort($array)

Ordena os valores do array. int count($array)

Retorna a quantidade de elementos do array.

Funções Nativas

string date($formato)

Retorna a data atual. Exemplo: 'd/m/Y H:i:s' void header($cabecalho)

Envia um cabeçalho HTTP. Exemplo: 'Location: /pagina.php'
float ceil($numero) | float floor($numero)

Arredonda para cima. | Arredonda para baixo. mixed min($array) | mixed min($numero1, $numero2, ...)

Retorna o menor valor. Também existe "max". int rand($min, $max)

Retorna um número aleatório no intervalo [$min, $max].
float round($numero[, $precisao])

Arredonda o número.

Funções Nativas

void exit

Interrompe a execução e envia para o cliente o que tiver no buffer. string trim($texto)

Retorna uma string sem os espaços do começo e do final. int strpos($texto, $palavra)

Retorna a posição da palavra no texto. Se não encontra retorna false. Usar ===.

string substr($texto, $comeco[, $tamanho])

Retorna um pedaço da string.

mixed str_replace($palavraAntiga, $palavraNova, $texto) Substitui as palavras antigas pelas novas e retorna o resultado.

Reuso

Conforme o seu programa cresce, o seu arquivo PHP também cresce. Precisamos organizar melhor o nosso programa. Vamos começar a separar em vários arquivos a nossa página.

Em um arquivo vamos colocar as nossas funções. Assim, quando o usuário acessar uma página, nós podemos incluir esse arquivo, assim, incluindo automaticamente as nossas funções.

Separar o programa em arquivos é essencial para uma boa organização no projeto. Além de incluir funções e classes PHP, também podemos incluir HTML. Pense em um menu que irá aparecer em todas as páginas. Invés de refazer o menu em cada página, podemos simplesmente incluir o arquivo do menu.

Reuso

Existem 4 comandos que podem ser utilizados para incluir um arquivo. Os detalhes de cada comando são exibidos na tabela.

Comando	Descrição
include	Em caso de erro, continua a execução.
require	Em caso de erro, aborta a execução.
include_once	Igual ao include, mas inclui no máximo 1 vez o arquivo.
require_once	Igual ao require, mas inclui no máximo 1 vez o arquivo.

Reuso

Exemplo de utilização do **include** em que o arquivo não existe.

No log de erros do Apache, irá aparecer *PHP Warning: include(): Failed opening 'erro' for inclusion*

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
<h3>Usando include em um arquivo inválido</h3>
<?php
include 'erro';
?>
<p>O texto aparece, pois a execução continua.</p>
</body>
</html>
```

Reuso

Vamos incluir uma página que possui o cabeçalho do site. Assim, quando eu precisar alterar o cabeçalho, eu altero somente um arquivo, e todas as páginas estarão atualizadas.

Perceba que o arquivo de cabeçalho só tem HTML. Para usar PHP, precisa usar as tags PHP também!

```php
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
    <?php include 'include2_cabecalho.php'; ?>
    <p>Corpo da página</p>
</body>
</html>
```

```php
<h3>Minha página pessoal</h3>
<ul>
    <li>Home</li>
    <li>Notícias</li>
    <li>Fotos</li>
</ul>
```

Reuso

Vamos supor que em um arquivo estão as funções essenciais para o nosso programa. Portanto, caso não carregue esse arquivo, então a execução deve ser abortada. Vamos incluir o arquivo com o comando **require**.

```php
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
<h3>Jogo da loteria</h3>
<p>
    <?php
    require 'funcoes_basicas.php';
    imprimirNumerosLoteria();
    ?>
</p>
</body>
</html>
```


```php
<?
function imprimirNumerosLoteria() {
    for ($numero = 0; $numero < 6; $numero++) {
        echo rand(1,60);
        if ($numero < 5) {
            echo ', ';
        }
    }
}
?>
```

Reuso

E se o arquivo passado para o comando require não existisse? Aconteceria um erro e a execução seria abortada. O cliente recebe o que estava no buffer de saída até o momento do erro.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
<h3>Usando require em um arquivo inválido</h3>
<p>Este texto é enviado para o cliente.</p>
<?php
require 'erro';
?>
<p>Este texto NÃO aparece, pois a execução é abortada.</p>
</body>
</html>
```

Reuso

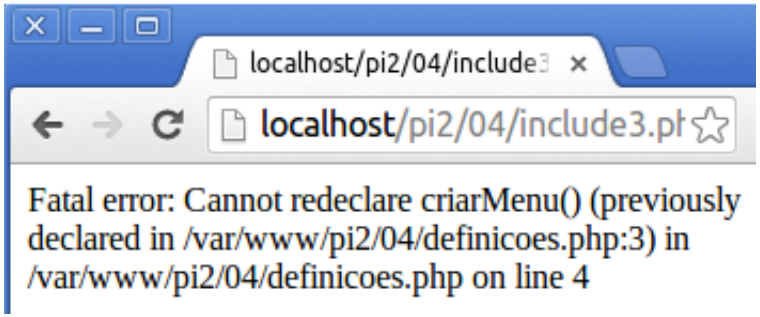
O log de erros do servidor é muito importante. Ele deve ser visto todos os dias em um servidor de produção. No servidor de desenvolvimento, podemos ativar a opção para mostrar erros direto no navegador. No arquivo de configuração do PHP (*php.ini*), existe uma opção chamada **display_errors**. Basta setar o valor para "On", que o erro é enviado para o cliente no meio do HTML.

Reuso

Lembrese que caso o arquivo incluído tenha definições de funções ou classes, nós só podemos incluílos uma vez. Se tentarmos incluir um arquivo desses 2 vezes, acontecerá um erro, pois já existirá uma função ou classe com tal nome declarada. Neste caso utilizamos a versão "_once" dos comandos include e require.

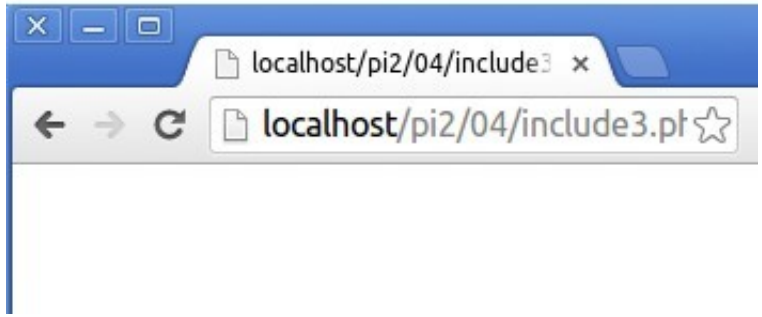
```php
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
    <?php include 'definicoes.php'; ?>
    <?php include 'definicoes.php'; ?>
</body>
</html>
```

```php
<?php
function criarMenu() {
    echo "menu criado";
}
?>
```




```php
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
    <?php include_once 'definicoes.php'; ?>
    <?php include_once 'definicoes.php'; ?>
</body>
</html>
```

```php
<?php
function criarMenu() {
    echo "menu criado";
}
?>
```



Referências

NIEDERAUER, Juliano. **Desenvolvendo websites com PHP**. 2. ed. São Paulo: Novatec, 2011. 301 p. ISBN 9788575222348.