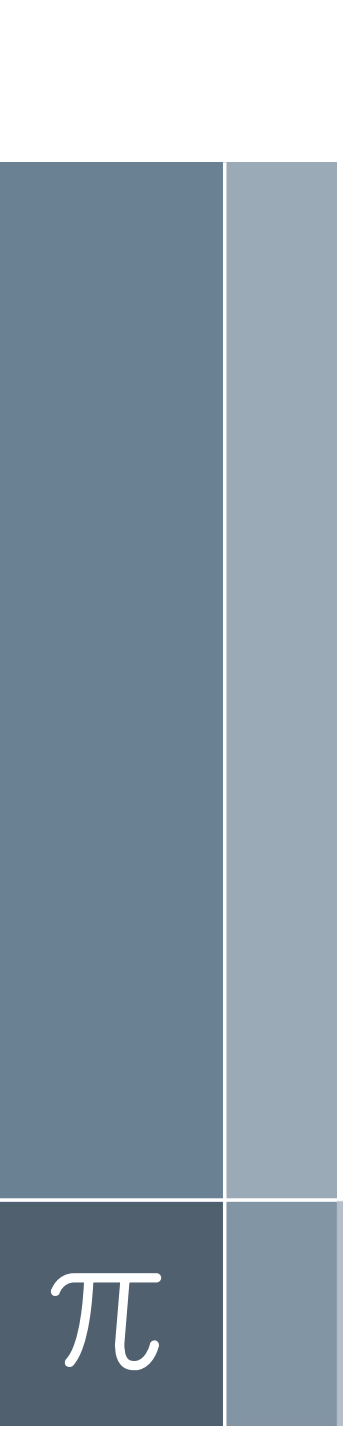
Linguagem de Programação

Aprendendo PHP



O que será abordado

- O que é a linguagem PHP
- Aprendendo sobre Variáveis e Constantes
- Aprendendo sobre vetor (array)
- Uso de operadores matemáticos e lógicos
- Comandos da linguagem (comandos repetição e comandos de decisão)
- Funções definidas pelo usuário-desenvolvedor
- POO em PHP

O que é a linguagem PHP

- O PHP pode ser obtido no site https://www.php.net
- PHP, que significa "PHP: Hypertext Preprocessor", é uma linguagem de programação de ampla utilização, interpretada (PHP, Manual)
- É uma linguagem interpretada, com sintaxe semelhante a linguagem C, Java e Perl
- Ela é executado no servidor.
- O PHP funciona como um programa CGI, a diferença está no fato do PHP ficar embutido no código HTML, diferente de programas CGI que precisam escrever todo o código HTML.

Suportes do PHP

- PHP suporta diversos sistemas de banco de dados:
  - dBase, Interbase, mSQL, mySQL,Oracle, Sybase, PostgreSQL, MSSQLServer entre outros
- Além do protocolo HTTP, o PHP suporta os protocolos:
  - IMAP, SNMP, NNTP,POP3.
  - Também é possível abrir *sockets* para interagir com outros protocolos.

Aprendendo sobre Variáveis e Constantes

- O PHP é *"case sensitive"*, ou seja, ele faz diferença entre caracteres maiúsculo e minúsculo quando você nomeia as variáveis e constantes.
- É recomendado escrever as variáveis com nomes em letras minúsculas, para evitar problemas com variáveis pré-definidas no PHP que são nomeadas usando caracteres em maiúsculo.
- No PHP as variáveis não são fortemente tipadas, sendo que uma variável pode mudar seu tipo de dados durante a execução do programa, passando de texto para número, de número para data, de data para objetos, de objeto para número em ponto flutuante, deste para array. Não existe uma ordem para a mudança.
- Toda variável ou constante possui o caracter $ no início do nome, seguido ou não de sublinhado (_), seguido de uma letra.
  - Ex: $_cidade ou $cidade

Variável Global do PHP

- $_REQUEST – Array contendo os parâmetros passados para o programa PHP.

  Abrange método Post, GET, Cookies
- Sintaxe: $_REQUEST[“nome do atributo”]

Aprendendo sobre Variáveis e Constantes

- Tipo Inteiro (integer ou long)
  - Uma variável que armazena um valor inteiro.
    - Quando o valor armazenado iniciar com o sinal de subtração (-) ou com um número entre 1 e 9, o PHP usará a base decimal.
    - Quando o valor iniciar com o valor 0, será utilizada a base octal.
    - Quando o valor inicial for 0x, será utilizada a base hexadecimal.
  - Existe o número inteiro longo chamado long, mas é o interpretador PHP que decide se será utilizado ou não, para isto ele considera o valor que está manipulando para determinar a quantidade de bytes necessários para armazena-lo.

Aprendendo sobre Variáveis e Constantes

- Tipo número em Ponto Flutuante (double ou float)
  - Representam os número não inteiros e são identificados pelo interpretador do PHP quando é utilizado o ponto decimal (.) ou a letra “e” no conteúdo do número.
    - Ex: 234.56 (duzentos e trinta e quatro inteiros e cinquenta e seis centésimos) ;
    - 23e4 que é igual a $23\ X\ 10^4\ = 23\ X\ 10.000 = 230.000$
    - Os números em ponto flutuante podem ser negativos com o sinal (-) no início do número ou positivos, sem o uso do sinal (-).
  - A notação científica de um número (usando o caracter e) pode indicar uma fração quando o valor após o (e) for negativo.
    - Ex: 23e-4 que é igual a $23\ X\ 10^{-4}\ = \frac{23}{10^4} = \frac{23}{10.000} = 0{,}0023$

Aprendendo sobre Variáveis e Constantes

- Tipo Texto (String)
  - Para indicar um string, o conteúdo deve aparecer em aspas (“) ou entre apóstrofes (‘).
  - A diferença esta como o interpretador PHP trata esta duas formas.
  - Quando usando aspas, o interpretador PHP processa qualquer variável e caracter de escape existente no string antes de ser atribuído.
  - Quando é utilizado apóstrofe, o interpretador PHP atribuirá exatamente o que esta na string, ou seja não fará o processamento de variáveis ou caracteres de escape.

Aprendendo sobre Variáveis e Constantes

- Tipo Vetor (array)
  - "Um array no PHP é na verdade um mapa ordenado. Um mapa é um tipo que relaciona valores a chaves. Este tipo é otimizado para várias usos diferentes: ele pode ser tratado como um array, uma lista (vetor), hashtable (que é uma implementação de mapa), dicionário, coleção, pilha, fila e provavelmente mais. Assim como existe a possibilidade dos valores do array serem outros arrays, árvores e arrays multidimensionais."(PHP, Manual)
  - As chaves quando não indicadas, são definidas pelo interpretador PHP iniciando com o valor 0.
  - A chave quando indicada um valor inteiro inicial, fará com que o interpretado PHP use o próximo valor inteiro maior se o chave não for indicada.

Aprendendo sobre Variáveis e Constantes

- Tipos para a chave do índice do vetor (coerção).
  - Se utilizado um string não numérico, este será o valor da chave.
  - Se utilizado um string numérico, ex: “25”, este será convertido no número 25.
  - Se utilizado um string numérico, ex: ”08”, este não será convertido, pois o valor 08 não é um número decimal e sim um número octal, como indicado no tipo Inteiro anteriormente.
  - Se utilizado um numero float, ex: 1.8, apenas a parte inteira será utilizada.
  - Se utilizado um booleano, true será convertido para 1 e false para 0.
  - Se utilizado o valor NULL, será convertido para uma string vazia “”.
  - Se utilizado um array, um erro será retornado, “Illegal offset type”

Aprendendo sobre Variáveis e Constantes

- Tipo Booleano (boolean)
  - São dois os valores possíveis para este tipo.
    - True para verdadeiro, sendo representado também por 1 nas coerções.
    - False para falso sendo representado também por 0 nas coerções.
  - Ao converter para booleano, os seguintes valores são considerados FALSE:
    - o próprio booleano FALSE
    - o inteiro 0 (zero)
    - o ponto flutuante 0.0 (zero)
    - uma string vazia e a string "0"
    - um array sem elementos
    - um objeto sem variáveis membros (somente PHP 4)
    - o tipo especial NULL (incluindo variáveis não definidas)
    - o objeto SimpleXML criado de tags vazias
  - Qualquer outro valor é considerado TRUE (incluindo qualquer recurso)

Aprendendo sobre Variáveis e Constantes

- "Uma constante é um identificador (nome) para um valor único. Como o nome sugere, esse valor não pode mudar durante a execução do script" (PHP,Manual)
- "O nome de uma constante tem as mesmas regras de qualquer rótulo do PHP. Um nome válido de constante começa com uma letra ou sublinhado, seguido por qualquer número de letras, números ou sublinhados" (PHP, Manual)
- Define é o comando utilizado para definir constantes
  - define("cidade", "Bragança Paulista");
  - const ESTADO = "São Paulo";
- "Ao contrário das funções definidas através de define(), as constantes definidas usando a palavra-chave const devem ser declarados no escopo de maior nível, pois são definidas no tempo de compilação. Isso significa que não podem ser definidas dentro de funções, laços ou instruções ifs ou blocos try/catch." (PHP,Manual)
- "Constantes definidas usando a palavra-chave const sempre serão *"case sensitive"*, enquanto as constantes usando define() podem ser *"case insensitive"*." (PHP,Manual)

Operadores

- Aritméticos
  - Adição (+) => $a = 5+3 // variável $a receberá o valor 8
  - Subtração (-) => $a = 5-3 // variável $a receberá o valor 2
  - Multiplicação (*) => $a = 5*3 // variável $a receberá o valor 15
  - Divisão (/) => $a = 5/3 //variável $a receberá o valor 1.66667
  - Módulo (%) => $a = 5%3 // variável $a receberá o valor 2, //resto da divisão de 5/3

Operadores

- String
  - Concatenação de string (.), usando o caracter ponto
  - Exemplo
    - $a = “Hoje está “; // string terminado com um espaço em branco para efeito da mensagem final
    - $b = “ensolarado”;
    - $c = $a . $b; // $c conterá a frase “Hoje está ensolarado”

Operadores

- Atribuição
  - (=) atribuição simples
    - $a = 10; // 10 está armazenado em $a
  - (+=) Atribuição com adição
    - $a += 3; // 3 será somado a $a que passa a armazenar 13.
  - (-=) Atribuição com subtração
    - $a -= 4; // 4 será subtraído de $a que passa a armazenar 9.
  - (*=) Atribuição com multiplicação
    - $a *= 2; // 2 será multiplicado por $a que passa a armazenar 18.
  - (/=) Atribuição com divisão
    - $a /= 3; // 3 será divisor de $a que passa a armazenar 6.
  - (%=) Atribuição com módulo
    - $a %= 5; // 5 será divisor de $a que passa a armazenar 1 como resto da divisão.
  - (.=) Atribuição com concatenação
  - $a .= “ é resto da divisão”; // $a passa a armazenar a string “1 é o resto da divisão”

Operadores

- Atribuição
  - (++) Incremento
    - $a++ ; // O incremento é realizado no final do comando, se uma //variável receber o valor, o valor recebido será anterior ao //incremento. Ex $b = $a++, $b terá o valor antes do incremento // de $a
    - ++$a; // O incremento é realizado no inicio do comando, se uma //variável receber o valor, o valor recebido será igual ao //incremento. Ex $b = ++$a, $b terá o valor igual do incremento de $a
  - (--) Decremento
    - $a-- ; // O decremento é realizado no final do comando, se uma //variável receber o valor, o valor recebido será anterior ao //decremento. Ex $b = $a--, $b terá o valor antes do decremento // de $a
    - ++$a; // O decremento é realizado no inicio do comando, se uma //variável receber o valor, o valor recebido será igual ao //decremento. Ex $b = --$a, $b terá o valor igual do idecremento // de $a

Operadores

- Bit a bit – compara dois números bit a bit.
  - (&) “e” lógico
  - (|) “ou” lógico
  - (^) “ou exclusivo
  - (~) não (inversor)
  - (<<) deslocamento esquerda (multiplica por 2)
  - (>>) deslocamento direita (divide por 2)

  - Exemplos serão colocados em um arquivo php.

Operadores

- Lógicos – Usados para valores booleanos
  - (and) – “e” lógico
  - (or) – “ou” lógico
  - (xor) – “ou exclusivo
  - (!) – não (inversão)
  - (&&) – “e” lógico
  - (||) – ”ou” lógico

Operadores

- Comparação – Compara valores de variáveis e retorno um booleano
  - (==) – igual a
  - (!=) - diferente de
  - (<) – menor que
  - (>) – maior que
  - (<=) – menor ou igual a
  - (>=) maior ou igual a

Comando echo

- Exibe uma ou mais string.
- Sintaxe:
- echo (arg1, arg2, ...,arg3);
- Forma abreviada
- <?= string? >
- Obs. Echo não é uma função, por isto os argumentos não precisam ser passados dentro dos parênteses.

Comandos - IF

- O comando IF é utilizado para a avaliação de uma ou muitas condições, o resultado da avaliação das condições desvia a execução do programa para o bloco “then/então”, não representado no comando, ou para o bloco “else/senão”, representado por else.
- O Bloco else é opcional. Use “{“ e “}” para delimitar os blocos de comandos, caso contrário apenas um comando é executado.
- Sintaxe:
  - if ((condição1) [<operador lógico> (condição2)])
    - {Bloco quando a avaliação é verdadeira}
    - else
    - {Bloco quando a avaliação é falsa};

Comando - IF

- O PHP fornece a estrutura “elseif” que pode ser inserida no comando if, mas ela deve ser utilizada antes do “else”.
- Não há limites para utilizar “elseif”.
- O “elseif” exigi o teste lógico, como o comando IF.

Comando Switch

- “A declaração switch é similar a uma série de declarações IF na mesma expressão. Em muitos casos, se deseja comparar as mesmas variáveis (ou expressões), com diferentes valores, e executar pedaços diferentes de código dependendo de qual valor ela é igual.
  Esta é exatamente a serventia da declaração switch.” (PHP,Manual)
- Sintaxe:

```
switch (variável) {
        case valor1:
                bloco de comandos;
                break;
        case valor2:
                bloco de comandos;
                break;
        case valor3:
                bloco de comandos;
                break;
       default:
                bloco de comandos;
}
```

Comando While

- “O comando ‘while’ executa um bloco de comandos, até que uma condição lógica se torne falsa.” (PHP, Manual)
- Sintaxe:
  - while ((condição1) [<operador lógico> (condição2))
    - {bloco de comandos}
- Sintaxe alternativa:
  - while ((condição1) [<operador lógico> (condição2)) :
    - Bloco de comandos sem o uso das chaves { }
  - endwhile.
  - Obs. Note o dois pontos (:) no final da declaração do while, isto habilita o uso do endwhile e elimina as chaves para determinar o bloco de comandos que será executado pelo while.

Comando Do...While

- “Semelhando ao comando While, o Do... While executa um bloco de comando até a condição se tornar falsa. A diferença é que o teste da condição ocorre no final do processamento, com isto pelo menos uma vez o bloco é executado.” (PHP,Manual)
- Sintaxe:
  - do
  - { bloco de comandos }
  - while ((condição1) [<operador lógico> (condição2))

Comando FOR

- “O comando FOR executa a repetição de blocos de comandos até que uma condição se torne falsa. Ele é mais utilizado quando sabe-se antecipadamente quantas execuções acontecerão.” (PHP,Manual)
- Sintaxe:
  - for (expr1 ; expr2; expr3)
  - { bloco de código }
- Obs: expr1 é avaliada no inicio da execução e apenas uma vez;
- expr2 é avaliada a cada repetição executada e termina a repetição quando se torna FALSE
- Expr3 é executada no final de cada repetição

Comando FOREACH

- “O construtor foreach fornece uma maneira fácil de iterar sobre arrays. O foreach funciona somente em arrays e objetos, e emitirá um erro ao tentar usá-lo em uma variável com um tipo de dado diferente ou em uma variável não inicializada.” (PHP,Manual)
- Possui duas sintaxes:
  - foreach (array_expression as $value)
    - { BLOCO DE COMANDOS; }
  - foreach (array_expression as $key => $value)
    - { BLOCO DE COMANDOS; }

Função Print()

- Exibe apenas uma string.
- Sintaxe:
- print(arg);
- Obs. Print aceita apenas um argumento e sempre devolve o valor 1.

Funções definidas pelo usuário-desenvolvedor

- Uma função pode ser definida usando a seguinte sintaxe:
  - function nome_da_funcao ($arg1, $arg2, $arg3, /* ..., */$arg_n)
  - { bloco de código;
  - return $valor_para_retorno;
  - }
- O PHP suporta funções anônimas, estas funções são criadas como a função acima porém sem um nome.
  - function ($arg1,$arg2,$arg3,/* ..., */$arg_n)
  - { bloco de código;
  - return $valor_para_retorno;
  - }

Funções definidas pelo usuário-desenvolvedor

- Argumentos de funções
  - “Informações podem ser passadas para funções através da lista de argumentos, que é uma lista de expressões delimitados por vírgulas. Os argumentos são avaliados da esquerda para a direita.
    O PHP suporta a passagem de argumentos por valor (o padrão), passagem por referência, e valores padrões de argumentos. lista de argumentos de tamanho variável também são suportadas.” (PHP,Manual)
  - A passagem por valor é reconhecido por um argumento definido como uma variável comum. Ex: $nome_argumento;
  - A passagem por referência é reconhecido por um argumento definido como uma variável comum, porém antecedida por &. Ex: &$nome_de_outro_argumento;
  - A lista de argumentos é reconhecido pelo uso de 3 pontos (...) antes do argumento. Ex: ...$lista_argumentos_de_tamanno_variável;”

Manipulação de Vetor

- array() - cria um vetor vazio
- array(“nome” => “valor”) – cria um vetor onde o primeiro elemento é determinado por “nome” e este possuirá um valor determinado em “valor”.
- array_push($”nome do vetor”, “valor”) – acrescenta no vetor determinado por $”nome do vetor” o conteúdo definido em “valor”, este conteúdo pode ser um tipo primitivo, um vetor ou outro tipo de dado.
- array_pop($”nome do vetor”) — Extrai um elemento do final do vetor.
- array_shift($”nome do vetor”) — Retira o primeiro elemento de um vetor.
- array_unshift($”nome do vetor”) - Adiciona um ou mais elementos no início de um vetor.

Comandos para gerenciamento de sessão

- Sessão é um local para armazenar dados temporariamente no Servidor.
- session_start() — Inicia uma nova sessão ou resume uma sessão existente.
- session_abort() — Descarta as alterações no vetor da sessão e encerra a sessão.
- session_unset() — Libera todas as variáveis de sessão.
- $_SESSION - Um vetor associativo contendo variáveis de sessão disponíveis para o atual script.

Redirecionamento de página

- header - A função header () envia um cabeçalho HTTP simples para um cliente.
  - header(*header*, *replace*, *http_response_code*)

Parâmetro	Descrição
header	Obrigatório. Especifica o texto que será enviado.
replace	Opcional. Indica quando o cabeçalho pode substituir um cabeçalho enviado previamente ou adiciona um novo cabeçalho do mesmo tipo. O padrão é TRUE (com substituição) . False permite múltiplos cabeçalhos de mesmo tipo.
http_response_code	Opcional. Força o retorno de um código específico para a resposta do HTTP.

Acesso a Banco de Dados.

- PDO - utilizado para instanciar um objeto que acessa o banco de dados.
  - Ex.
```
try {
    $dbh = new PDO($dsn, $user, $password);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}
```
- Query – comando utilizado para realizar consulta ao banco de dados
  - Ex.
```
$sql = 'SELECT name, color, calories FROM fruit ORDER BY n
ame';
foreach ($conn->query($sql) as $row) {
    print $row['name'] . "\t";
    print $row['color'] . "\t";
    print $row['calories'] . "\n";
}
```

π

- Execute – comando utilizado para realizar alteração no banco de dados
  - Ex:

```
$calories = 150;
$colour = 'gre';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour LIKE :colour');
$sth->bindParam(':calories', $calories, PDO::PARAM_INT);
$sth->bindValue(':colour', "%{$colour}%");
$sth->execute();
```

- beginTransaction – Utilizado para iniciar uma transação atómica
  - EX: `$dbh->beginTransaction();`

```
/* Insert multiple records on an all-or-nothing basis */
$sql = 'INSERT INTO fruit
    (name, colour, calories)
    VALUES (?, ?, ?)';

$sth = $dbh->prepare($sql);

foreach ($fruits as $fruit) {
    $sth->execute(array(
        $fruit->name,
        $fruit->colour,
        $fruit->calories,
    ));
}

/* Commit the changes */
$dbh->commit();
```

- Commit – utilizado para efetivar as alterações pendentes em uma transação atómica

π

- Rollback – utilizado para desfazer uma transação que não foi confirmada.
  - Ex: `$dbh->beginTransaction();`

```
/* Change the database schema and data */
$sth = $dbh->exec("DROP TABLE fruit");
$sth = $dbh->exec("UPDATE dessert
    SET name = 'hamburger'");

/* Recognize mistake and roll back changes */
$dbh->rollBack();
```

- Prepare – utilizado para enviar um comando SQL parametrizado ao banco de dados
  - Ex:

```
$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth = $dbh->prepare
   ($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$red = $sth->fetchAll();
$sth->execute(array(':calories' => 175, ':colour' => 'yellow'));
```

- bindParam – utilizado para alterar valores de parâmetros em um sql.
  - Ex.:

```
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindParam(':calories', $calories, PDO::PARAM_INT);
$sth->bindParam(':colour', $colour, PDO::PARAM_STR, 12);
$sth->execute();
```

- BindValue – semelhante a bindParam.
  - EX.:

```
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour');
$sth->bindValue(':calories', $calories, PDO::PARAM_INT);
$sth->bindValue(':colour', $colour, PDO::PARAM_STR);
$sth->execute();
```

Referencias

- PHP, Manual. site:https://www.php.net/manual/pt_BR/preface.php. Último acesso: 07 Abr. 2019
- Curso de Linguagem PHP. Barreto, Maurício Vivas de Souza. CIPSGA. Comitê de Incentivo a Produção de Software Gratuito e Alternativo. Abr. 2000