Linguagem de Programação II

Prof.ª Ma. Jessica Oliveira

FBr Faculdade Brasília

Relembrando...

- **Vocês trabalharão em trios**, então escolham seus colegas sabendo que caminharão juntos até o fim do semestre;
- Todas entregas serão realizada pelo **AVA**, no máximo até às 23h59min do dia posterior;
- Todos da equipe deverão realizar a submissão, tanto do **arquivo .zip** contendo o que foi desenvolvido na aula, quanto do **relatório.**

Relembrando...

- Cada entrega dos microprojetos (4, por bimestre), com seu respectivo relatório, valerá **ATÉ** 0,05 ponto, o que totaliza **ATÉ** 2,00 pontos por bimestre;
- O relatório parcial, que deverá ser entregue e apresentado no dia da revisão para AV1, valerá **ATÉ** 2,00 pontos;
- O relatório final, junto da apresentação e entrega final, valerá **ATÉ** 2,00 pontos, e será realizada na semana anterior à revisão para AV2.

Relembrando...

- PHP é compilado ou interpretado?
- Qual a *tag* padrão para delimitar o código PHP?
- O que eu JAMAIS posso esquecer de colocar após cada instrução no PHP?
- Quais tipos de dados eu tenho no PHP?
- Posso usar qualquer tipo de aspas nas variáveis?
- PHP diferencia letra maiúscula de minúscula?
- O que são escopos? Quais são os tipos?
- O que é uma constante?

Aula 04 – 26/08/2024

Estruturas de Controle e Funções

O que são Expressões e Operadores?

- Uma **expressão** é uma combinação de valores, variáveis e operadores que o PHP avalia para produzir um resultado.
- Os **operadores** são símbolos que instruem o PHP sobre o que fazer com os valores e variáveis em uma expressão.

Operadores Aritméticos

- Usados para realizar operações matemáticas básicas.
- **Exemplos:**

```
$soma = 5 + 3; // Resultado: 8
$subtracao = 5 - 3; // Resultado: 2
$multiplicacao = 5 * 3; // Resultado: 15
$divisao = 6 / 3; // Resultado: 2
```

Operadores Lógicos

- Usados para combinar expressões *booleanas* e retornar verdadeiro ou falso.
- **Exemplos:**

```
$a = true && false; // Resultado: false
$b = true || false; // Resultado: true
$c = !true; // Resultado: false
```

Operadores Relacionais

- Comparam dois valores e retornam verdadeiro ou falso.
- **Exemplos:**

```
$maior = 5 > 3; // Resultado: true
$igual = 5 == 3; // Resultado: false
$diferente = 5 != 3; // Resultado: true
```

Testes Condicionais: *If-Else*

- Permitem que nosso código tome decisões com base em condições.
- O ***if*** executa um bloco de código se a condição for **verdadeira**.
- O ***else*** executa um bloco de código se a condição for **falsa**.

Testes Condicionais: *If-Else*

- **Exemplo:**

```
$idade = 18;
if ($idade >= 18) {
    echo "Você é maior de idade.";
} else {
    echo "Você é menor de idade.";
}
```

Testes Condicionais: *Switch*

- É uma alternativa ao *if-else* para quando precisamos verificar o valor de uma variável contra diferentes casos.
- **Exemplo:**

```php
$opcao = 2;
switch ($opcao) {
    case 1:
        echo "Opção 1 selecionada.";
        break;
    case 2:
        echo "Opção 2 selecionada.";
        break;
    default:
        echo "Opção inválida.";
}
```

Comandos de Repetição: *For*

- É usado para repetir um bloco de código um número definido de vezes.
- **Exemplo:**

```
for ($i = 1; $i <= 5; $i++) {
    echo "Número: " . $i;
}
```

Comandos de Repetição: *While*

- Executa um bloco de código enquanto a condição for verdadeira.
- **Exemplo:**

```
$i = 1;
while ($i <= 5) {
    echo "Número: " . $i;
    $i++;
}
```

Comandos de Repetição: *Do-While*

- É semelhante ao *while*, mas garante que o código seja executado pelo menos uma vez.
- **Exemplo:**

```
$i = 1;
do {
    echo "Número: " . $i;
    $i++;
} while ($i <= 5);
```

O que são Funções?

- São **blocos de código** que podem ser chamados várias vezes no programa, permitindo a **reutilização** de código.
- Elas ajudam a **modularizar** e **organizar** o código, tornando-o mais **legível** e **fácil de manter**.

Como declarar Funções?

- Uma função em PHP é declarada com a palavra-chave *function*, seguida pelo nome da função e um bloco de código.
- **Exemplo:**

```
function saudacao() {
    echo "Olá, humano!";
}
saudacao(); // Chama a função e exibe "Olá, humano!"
```

Escopo de Variáveis em Funções

- As variáveis declaradas dentro de uma função têm escopo local, ou seja, só existem dentro daquela função.
- Variáveis globais, declaradas fora de qualquer função, podem ser acessadas em qualquer parte do código.
- **Exemplo:**

```php
$mensagem = "Olá, humano!"; // Variável global
function saudacao() {
    $mensagem = "Olá, ADS!"; // Variável local
    echo $mensagem;
}
saudacao(); // Exibe "Olá, ADS!"
echo $mensagem; // Exibe "Olá, humano!"
```

Passando Parâmetros para Funções

- Funções podem receber valores (parâmetros) para trabalhar com eles.
- Os valores passados para a função na hora de chamá-la são chamados de **argumentos**.
- **Exemplo:**

```
function soma($a, $b) {
    return $a + $b;
}
echo soma(2, 3); // Exibe 5
```

Retornando Valores de Funções

- Funções podem retornar valores usando a palavra-chave *return*.
- O valor retornado pode ser armazenado em uma variável ou usado diretamente.
- **Exemplo:**

```php
function maior($a, $b) {
    if ($a > $b) {
        return $a;
    } else {
        return $b;
    }
}
echo maior(10, 20); // Exibe 20
```

Vamos para a prática?

Não esqueçam, todos do trio devem realizar as entregas!

Microprojeto 1:
Estrutura básica do sistema

Passo 1: Criando a estrutura do projeto

- Primeiro, organize o seu ambiente de trabalho. Navegue até a pasta onde o XAMPP está instalado no seu computador. Normalmente, o caminho é algo como "`C:\xampp\`".
- Dentro da pasta "`htdocs`", crie uma nova pasta para o projeto, por exemplo, "`sistema_eventos`";
- Dentro da pasta "`sistema_eventos`", crie um arquivo PHP chamado "`index.php`", que será o ponto de entrada do sistema.

Passo 2: Definindo as ações do sistema

- Nosso sistema deve ser capaz de realizar diferentes ações, como criar um evento e visualizar todos os eventos cadastrados. Como ainda não estamos usando HTML para formular essas ações, vamos simular as entradas através de variáveis PHP.

Passo 3: Implementando a lógica de decisão com *If-Else*

- Agora, vamos usar uma estrutura de controle “`if-else`” para determinar qual ação deve ser executada com base no valor de “`$acao`”.

Passo 4: Criando funções para modularizar o código

- Vamos agora definir as funções “`criarEvento`” e “`visualizarEventos`”. Elas encapsulam a lógica de manipulação de dados, tornando o código mais organizado e fácil de manter.

Passo 5: Testando o Microprojeto

- Depois de implementar as funções e a lógica de decisão, podemos testar o sistema alterando o valor de **$acao** e observando os resultados.

Teste 1: Criar um evento

- **Código:** certifique-se de que `$acao = "criar";` e execute o código.

Teste 2: Visualizar eventos

- **Código:** mude `$acao = "visualizar";` e execute o código.

Na próxima aula...

Aula 05 (03/09/2024) - Introdução à Orientação a Objetos (OO).

Dúvidas?

jessica.oliveira@fbr.edu.br