

Anotações estudando o PHP em sala de Aula

Objetivo

vamos aprender sobre alguns conceitos fundamentais a respeito do php 5 para o desenvolvimento web

Tópicos Abordados:

- *Introdução ao PHP*
- *Sintaxe básica do PHP*
- *Variáveis GET e POST*
- *Sessões*
- *Conexão com o PostgreSQL*
- *Includes e require*

Introdução ao PHP

Antes de criarmos um exemplo de sintaxe em PHP, é necessário entender que o arquivo deve ter a extensão **.php** para que o servidor (como o XAMPP) reconheça que se trata de um script PHP.

Exemplo de nome de arquivo: **produtos.php**

O PHP utiliza uma tag de abertura e fechamento para delimitar o código. Todo o script PHP deve estar contido dentro dessas tags.

Exemplo:

```
<?php
```

```
    echo "Esse é o nosso programa PHP. Devemos respeitar essa estrutura.";
```

```
?>
```

Observações:

- A tag de abertura do PHP é: `<?php` e a de fechamento é: `?>`.

O PHP pode estar embutido em HTML (e vice-versa) - Uma das características mais poderosas do PHP é a possibilidade de integrá-lo diretamente em páginas HTML. Isso permite criar páginas dinâmicas de forma simples.

Você pode abrir um bloco PHP dentro de um arquivo **.php** que contenha código HTML.

Exemplo: abrindo o arquivo.php

```
<!DOCTYPE html>

<html>

    <head>

        <title>Exemplo com PHP</title>

    </head>

    <body>

        <h1>Bem-vindo!</h1>

        <?php

            echo "<p>Hoje é " . date("d/m/Y") . ".</p>";

        ?>

    </body>

</html>
```

Resumindo: Dentro de um arquivo.php podemos fazer um script html e podemos mesclar um bloco de código php neste script html

Html gerado pelo script php - Também é possível gerar trechos ou até mesmo uma página HTML completa utilizando apenas código PHP. Isso é feito através da função **echo**, **print**, ou mesmo estruturas de controle que imprimem HTML de forma condicional ou repetitiva.

Exemplo: Abrindo o arquivo.php

```
<?php

    echo "<!DOCTYPE html>";
    echo "<html>";
```

```
echo "<head><title>Página gerada em PHP</title></head>";
echo "<body>";
echo "<h1>Olá, mundo!</h1>";
echo "<p>Essa página foi gerada dinamicamente usando PHP.</p>";
echo "</body>";
echo "</html>";
?>
```

Ao estudar a arquitetura **CGI (Common Gateway Interface)**, é possível perceber que essa abordagem — gerar HTML a partir de scripts — é uma herança direta desse modelo. A diferença é que, com PHP, essa funcionalidade está **integrada ao servidor**, como o Apache com o módulo **mod_php**, o que oferece um desempenho melhor comparado à execução de scripts CGI tradicionais (como scripts Perl ou Python executados de forma isolada).

Comparativo com CGI:

- No CGI tradicional, cada requisição cria um novo processo para executar o script.
- Com o PHP integrado (ex: mod_php ou PHP-FPM), o processamento é feito internamente no servidor, sem a criação de processos externos, resultando em maior eficiência.

Apesar de possível, misturar muito PHP com HTML pode dificultar a manutenção. Em projetos maiores, é comum separar as responsabilidades utilizando templates (como o [Blade](#) no Laravel) ou engines como Twig.

SINTAXE básica DO PHP

Delimitação de código - Como vimos anteriormente o código é delimitado em um bloco contendo a tag `<?php>`, em que a parte de abertura do bloco de código é: `<?php` e a parte final é: `?>`

Instruções e ponto e vírgula - Como demonstrado na introdução a cada instrução/comando/script no bloco de código é necessário colocar um ponto e vírgula ; sempre!

Comentários - Para realizar comentários é bem simples, veja os exemplos abaixo:

```
<?php
```

```
// Comentário de uma linha
```

```
# Também é um comentário de uma linha
```

```
/*  
Comentário de múltiplas linhas  
*/
```

```
?>
```

Variáveis - As variáveis em PHP começam com o símbolo **\$** e não precisam de declaração de tipo, veja os exemplos abaixo:

```
<?php  
$nome = 'Guilherme Henrique Almeida da Silva';  
$saldo = 100.000;  
$status = true;  
  
if ($status == true) {  
    echo "<p>Olá, $nome. Seu saldo é: R$ $saldo</p>";  
}  
?  

```

observação: o ponto final é o operador de concatenação

Tipos de dados comuns:

- **string** (texto): "Olá"
- **int** (inteiro): 42
- **float** (decimal): 3.14
- **bool** (booleano): true ou false
- **array**: coleção de valores
- **object**: instância de uma classe
- **null**: valor nulo

Função echo (saída de dados) - Como vimos anteriormente essa função é a que exibe informações para o navegador.

Sensibilidade a maiúsculas e minúsculas - PHP é case sensitivity, ou seja, faz diferença de letras maiúsculas e minúsculas. Exemplo:

```
<?php  
$nome = "João";  
echo $nome; // Funciona
```

```
echo $NOME; // Erro: variável não definida
?>
```

Variáveis GET e POST:

definição resumida:

- **\$_GET** : Captura dados enviados via URL (query string)
- **\$_POST** : Captura dados enviados via formulário

Exemplo usando **\$_GET**: abrindo arquivo **ex.php**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo do uso do $_GET</title>
  </head>
  <body>
    <!-- URL: pagina.php?nome=Joao&idade=30 →

    <a href="pagina.php?nome=Joao&idade=30">Clique aqui</a>

    <!-- Comentário: Tudo depois do ? é uma requisição via GET ---->
    <!-- Comentário: o nome e idade é uma variável que comporta o
valor
    Para enviar via get e o & é o and na url →

    <?php
      if (isset($_GET['nome']) && isset($_GET['idade'])) {
        $nome = $_GET['nome'];
        $idade = $_GET['idade'];
        echo "<p>Nome: $nome</p>";
        echo "<p>Idade: $idade</p>";
      } else {
        echo "<p>Nenhum dado foi enviado via GET
ainda.</p>";
      }
    /*
```

Usei **isset()** para verificar se os parâmetros **nome** e **idade** foram enviados na URL, evitando warnings. Basicamente a função **isset()** verifica se uma variável foi definida e se o valor dela não é null. Retorna **true** se a variável existe e não é null **false** se a variável não foi definida

```

        ou é null.
    */
    ?>
</body>
</html>

```

Exemplo usando \$_POST: abrindo arquivo ex.php

```

<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo do uso do $_POST</title>
    </head>
    <body>

        <form action="recebe.php" method="POST">
            Nome: <input type="text" name="nome"><br>
            Idade: <input type="number" name="idade"><br>
            <input type="submit" value="Enviar">
        </form>

    </body>
</html>

```

abrindo arquivo receber.php:

```

<?php
    $nome = $_POST['nome'];
    $idade = $_POST['idade'];

    if (isset($_POST['nome']) && isset($_POST['idade'])) {
        $nome = $_POST['nome'];
        $idade = $_POST['idade'];

        echo "Nome: $nome <br>";
        echo "Idade: $idade";
    }else{

        echo "<p>Nenhum dado foi enviado via POST
        ainda.</p>";

    }
?>

```

Sessões

Uma **sessão** permite armazenar **informações específicas do usuário** entre diferentes requisições (páginas). Essas informações ficam **no servidor**, e um ID da sessão é mantido no navegador por meio de um **cookie** chamado

PHPSESSID

Basicamente uma sessão é uma mecanismo para armazenar dados entre requisições. **Observação - a sessão no php é um array associativo superglobal.**

Iniciando, criando variáveis de sessão e inserindo dados em uma sessão no php:

```
<?php
    session_start(); // inicia uma sessão

    if (isset($_POST['nome']) && isset($_POST['senha'])){

        // inserindo ou armazenando dados em uma sessão
        $_SESSION['username'] = $_POST['nome'];
        $_SESSION['password'] = $_POST['senha'];

    }
?>
```

Observações:

- **`$_SESSION['variável de sessão']` -> assim criamos a variável de sessão**

Acessando dados de uma sessão:

```
<?php
    session_start(); // Sempre necessário para acessar $_SESSION

    echo "<p>{$_SESSION['username']} bem-vindo</p>";
?>
```

Removendo dados de uma sessão:

```
<?php
    session_start(); // Sempre necessário para acessar $_SESSION

    echo "<p>{$_SESSION['username']} bem-vindo</p>";
    unset($_SESSION['username']); // remove apenas essa variável
?>
```

Destruindo toda a sessão:

```
<?php
    session_start(); // Sempre necessário para acessar $_SESSION

    echo "<p>{$_SESSION['username']} bem-vindo</p>";
    unset($_SESSION['username']); // remove apenas essa variável

    session_destroy(); // remove todos os dados da sessão
?>
```

observação: `session_destroy()` não remove os dados da superglobal `$_SESSION` imediatamente. Eles ainda existem no array enquanto o script atual estiver rodando.

Conexão com o Banco de Dados PostgreSQL:

No arquivo de configuração do xampp (php.ini) inclua a linha abaixo para ativar o PostgreSQL no Php5.

```
extension=php_pgsql.dll
```

Criar a conexão

```
<?php
    $conn = pg_connect("host=localhost dbname=produtos
    user=postgres password=123456");
    if (!$conn) {
        echo "Erro na conexão com o banco de dados.";
    }
?>
```

Executando uma Consulta

```
<?php
    $resultado = pg_query($conn, "SELECT * FROM usuarios");
    while ($linha = pg_fetch_assoc($resultado)) {
```



```
        echo "Nome: " . $linha['nome'] . "<br>";
    }
?>
```

Inserindo os dados

```
<?php
    $query = "INSERT INTO usuarios (nome, email) VALUES ('Carlos',
'carlos@email.com')";
    $resultado = pg_query($conn, $query);
    if ($resultado) {
        echo "Registro inserido com sucesso!";
    } else {
        echo "Erro ao inserir registro.";
    }
?>
```

Atualizando os dados

```
<?php
    $query = "UPDATE usuarios SET email = 'novoemail@email.com'
WHERE nome = 'Carlos'";
    $resultado = pg_query($conn, $query);
    if ($resultado) {
        echo "Registro atualizado com sucesso!";
    } else {
        echo "Erro ao atualizar registro.";
    }
?>
```

Fechando a conexão com o banco de dados:

```
<?php
    pg_close($conn);
?>
```

Include e Require

Tanto include quanto require são usados para inserir o conteúdo de um arquivo PHP dentro de outro. Isso é muito usado para cabeçalhos, rodapés, menus, conexões com banco de dados, etc.

Diferença entre um include e require

características	include	require
Erro ao não encontrar o arquivo	Emite um warning e continua o script	Emite um erro fatal e interrompe o script
Uso comum	Arquivos opcionais	Arquivos obrigatórios

Estrutura do exemplo

```
/meu_site/
|
|— index.php
|— sobre.php
|— contato.php
|
|— /includes/
|   |— header.php
|   |— footer.php
|   |— menu.php
```

Dentro do /includes/header.php:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <meta charset="UTF-8">
    <title>Meu Site com Include e Require</title>
  </head>
  <body>
```

Dentro do /includes/menu.php:

```
<nav>
  <a href="index.php">Início</a> |
```

```
        <a href="sobre.php">Sobre</a> |  
        <a href="contato.php">Contato</a>  
    </nav>  
<hr>
```

Dentro do /includes/footer.php:

```
<hr>  
    <footer>  
        <p>Site feito com PHP - Todos os direitos reservados © 2025</p>  
    </footer>  
</body>  
</html>
```

Index.php - Usando include

```
<?php  
    include 'includes/header.php'; // não é crítico — pode continuar se faltar  
    include 'includes/menu.php';   // idem  
?>
```

```
<h1>Página Inicial</h1>  
<p>Bem-vindo ao nosso site!</p>
```

```
<?php  
    include 'includes/footer.php';  
?>
```

Sobre.php - Usando require

```
<?php  
    require 'includes/header.php'; // obrigatório — se faltar, página quebra  
    require 'includes/menu.php';   // idem  
?>
```

```
<h1>Sobre Nós</h1>  
<p>Este é um site de exemplo criado para demonstrar o uso de include e  
require.</p>
```

```
<?php  
    require 'includes/footer.php';  
?>
```

Contato.php - Usando o include e o require

```
<?php
```

```
    // evita incluir duas vezes, se já foi incluído  
    include_once 'includes/header.php';
```

```
    require_once 'includes/menu.php'; // idem
```

```
/* você pode incluir esse menu de novo e ele será ignorado porque já foi  
carregado */
```

```
        require_once 'includes/menu.php';  
?>
```

```
<h1>Contato</h1>
```

```
<p>Entre em contato pelo email: contato@exemplo.com</p>
```

```
<?php
```

```
    include_once 'includes/footer.php';  
?>
```