Introdução a desenvolvimento de sistemas WEB

O que é PHP?

História e Propósito

PHP (PHP: Hypertext Preprocessor) foi criado em 1994 por Rasmus Lerdorf.

Inicialmente chamado de "Personal Home Page Tools", evoluiu para uma linguagem de programação completa.

PHP como linguagem server-side

O PHP é executado no servidor, não no navegador do cliente.

Gera HTML que é enviado ao cliente, que não tem acesso ao código PHP original.

Diferença entre front-end e back-end

Front-end: HTML, CSS, JavaScript (cliente)

Back-end: PHP, Python, Java, etc. (servidor)

Versões atuais e evolução

PHP 8.2 (atual) - Melhorias de desempenho e segurança

PHP 8.0 - JIT compiler, union types, attributes

PHP 7.0 - Aumento significativo de velocidade

Sites famosos feitos com PHP

- Facebook - Inicialmente construído em PHP
- Wikipedia - Usa MediaWiki (PHP)
- WordPress - CMS mais popular (PHP)
- Etsy - Marketplace global
- Mailchimp - Marketing por email

Criando o Primeiro Arquivo PHP

Estrutura básica do arquivo .php

Arquivos PHP têm a extensão **.php** e podem conter HTML, CSS, JavaScript e código PHP.

O código PHP é executado no servidor antes de enviar o resultado ao cliente.

Abertura e fechamento do script PHP

O código PHP é delimitado pelas tags:

```
<?php
 // Seu código PHP aqui
?>
```

Exibindo mensagens com echo

O comando **echo** é usado para exibir texto ou variáveis na página:

```
<?php
 echo "Olá, mundo!";
```

Passos para criar seu primeiro arquivo PHP

1. Crie um arquivo com extensão .php (ex: index.php)
2. Adicione as tags de abertura e fechamento PHP
3. Escreva seu código PHP entre as tags
4.

Sintaxe Básica do PHP

Comentários

```
// Comentário de uma linha
/* Comentário de múltiplas linhas */
```

echo e print

echo : Exibe uma ou mais strings

```
echo "Olá, mundo!";
```

Variáveis

Começam com o símbolo **$** e são case sensitive

```
$nome = "João";
$idade = 25;
echo "Meu nome é $nome e tenho $idade anos.";
```

Tipos de dados

- **String** : $texto = "Olá";
- **Integer** : $numero = 42;
- **Float** : $preco = 19.99;
- **Boolean** : $ativo = true;

Operadores e Estruturas de Controle

Operadores Aritméticos

Operador	Descrição	Exemplo
+	Adição	$a + $b
-	Subtração	$a - $b
*	Multiplicação	$a * $b
/	Divisão	$a / $b
%	Módulo	$a % $b

Operadores de Comparação

Operador	Descrição
==	Igual (valor)
===	Idêntico (valor e tipo)
!=	Diferente
<, >	Menor que, Maior que

Estruturas Condicionais

```
if ($idade >= 18) {
 echo "Maior de idade";
 }else {
 echo "Menor de idade";
 }
```

Estruturas de Repetição

```
for ($i = 0; $i < 5; $i++) {
```

PHP e HTML

Inserindo PHP em uma página HTML

O PHP pode ser inserido em qualquer parte de um documento HTML.
Você pode alternar entre HTML e PHP quantas vezes quiser.

```
<!DOCTYPE html>
<html>
<head><title>Minha Página</title></head>
<body>

<h1> Bem-vindo,  <?php  echo $nome;  ?></h1>

<?php
 $hora = date("H:i");
 echo "<p>Agora são $hora</p>";
?>

</body>
</html>
```

Enviando dados via formulário


```
// processa.php
<?php
 $nome = $_POST["nome"] ;
 echo "Olá, $nome!";
?>
```

Superglobais: $_GET e $_POST

GET Dados enviados pela URL

Boas Práticas Iniciais

Indentação e organização do código

Use indentação consistente (2 ou 4 espaços)
Mantenha blocos de código bem organizados

```
// Ruim
if($idade>=18){echo "Maior de idade";}else{echo "Menor de idade";}
```

```
// Bom
if ($idade >= 18) {
echo "Maior de idade";
} else {
echo "Menor de idade";
}
```

Comentários úteis

Documente funções e blocos complexos
Evite comentários óbvios ou redundantes

```
/**
* Calcula a idade baseada na data de nascimento
* @param string $data_nascimento Formato: YYYY-MM-DD
```

Separação de lógica e apresentação

Separe o processamento de dados da exibição
Use arquivos diferentes para funções e templates

Evite código PHP misturado com HTML

Evite alternar frequentemente entre PHP e HTML
Prefira templates ou funções de saída organizadas

```
// Evite isso
<div>
```

Exercício Prático

Atividade

Criar uma página com um formulário que recebe nome e idade e exibe uma mensagem personalizada usando PHP.

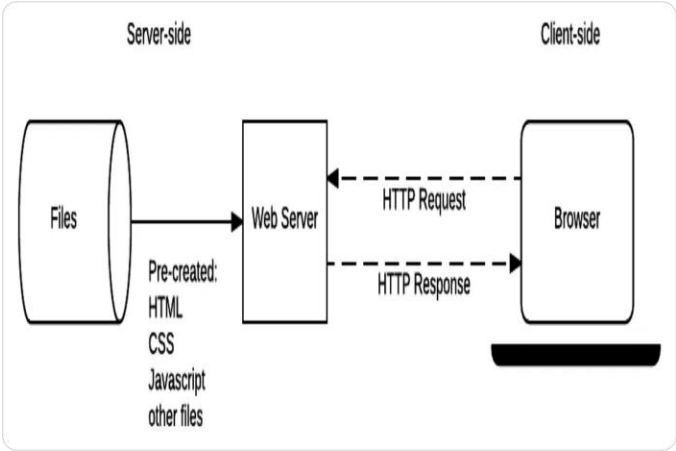
Formulário HTML (form.html)

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

Processamento PHP (processa.php)

```
<?php
 $nome = $_POST["nome"];
 $idade = $_POST["idade"];

 echo "Olá, $nome!";
```



Desafios adicionais

- Adicione validação para garantir que os campos não estejam vazios
- Crie uma mensagem personalizada baseada na faixa etária (criança, adolescente, adulto, idoso)
- Adicione CSS para melhorar a aparência do formulário e da página de resultado

Superglobais do PHP para Formulários

$_GET

Usado para capturar dados enviados via método GET.

```
$nome = $_GET['nome'];
```

Ver Exemplo

$_POST

Usado para capturar dados enviados via método POST (mais seguro que GET).

```
$email = $_POST['email'];
```

Ver Exemplo

$_REQUEST

Pode capturar dados enviados via GET, POST ou COOKIE.

```
$valor = $_REQUEST['valor'];
```

Ver Exemplo

Dica: Escolha o método adequado para cada situação

- Use GET para operações de leitura e parâmetros não sensíveis
- Use POST para envio de dados sensíveis ou formulários complexos
- Use REQUEST quando precisar acessar dados de diferentes origens

Validação e Sanitização

isset() e empty()

Verifica se uma variável está definida ou vazia.

```
if (isset($_POST['email'])) {
$email = $_POST['email'];
}
if (!empty($_POST['senha'])) {
$senha = $_POST['senha'];
}
```

filter_input()

Filtra e valida dados de entrada.

```
$email = filter_input(INPUT_POST, 'email',
FILTER_VALIDATE_EMAIL);
```

htmlspecialchars() e trim()

Evita XSS e remove espaços em branco.

```
$nome = htmlspecialchars($_POST['nome']);
$texto = trim($_POST['texto']);
```

intval() e floatval()

Converte entradas em números.

```
$idade = intval($_POST['idade']);
$preco = floatval($_POST['preco']);
```

Boas Práticas de Segurança:

- Sempre valide todos os dados de entrada do usuário
- Combine múltiplas funções para maior segurança

Upload de Arquivos

$_FILES

Usado para lidar com arquivos enviados via formulário.

```
$nomeArquivo = $_FILES['arquivo']['name'];
$caminhoTemporario = $_FILES['arquivo']['tmp_name'];
move_uploaded_file($caminhoTemporario, "uploads/$nomeArquivo");
```

Propriedades de $_FILES

- name : Nome original do arquivo
- tmp_name : Caminho temporário do arquivo
- size : Tamanho do arquivo em bytes
- type : Tipo MIME do arquivo
- error : Código de erro (se houver)

Exemplo de Formulário

```
<form method="post" enctype="multipart/form-data">
<input type="file" name="arquivo">
<button type="submit">Enviar</button>
</form>
```

Importante: Não esqueça do atributo enctype="multipart/form-data"