Programação para Internet

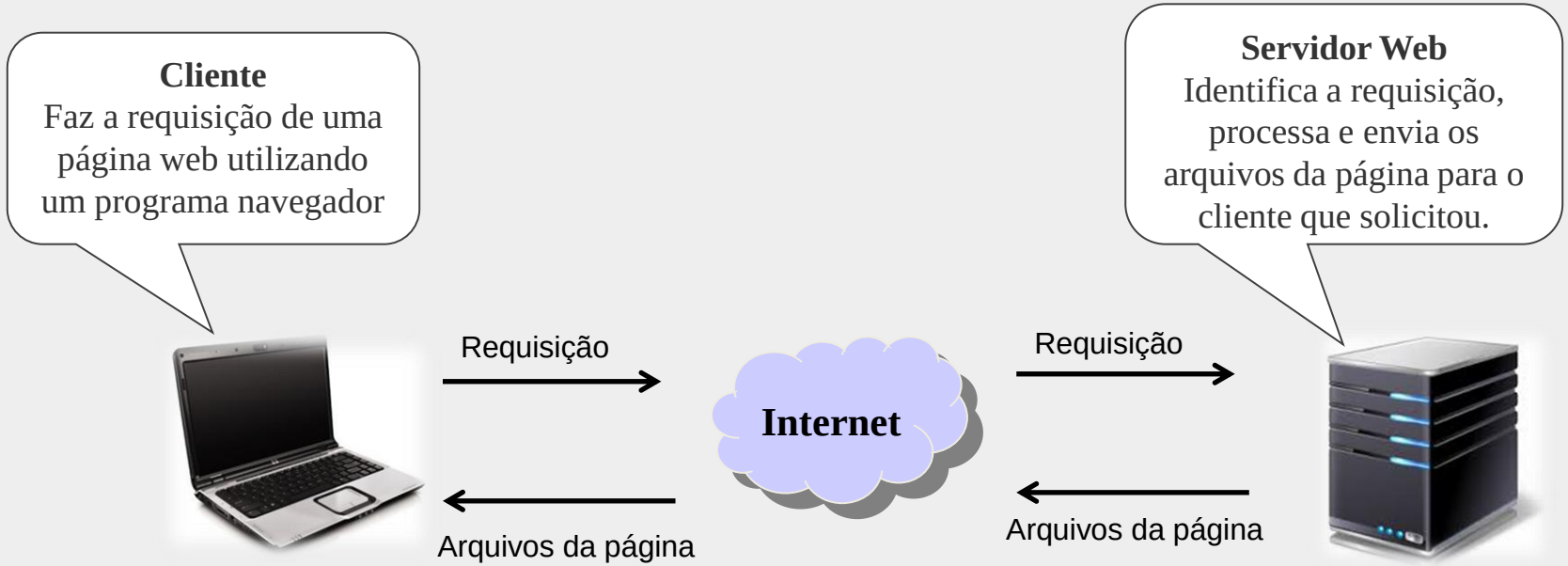
Módulo 5

Introdução à Web Dinâmica com PHP

Prof. Dr. Daniel A. Furtado - FACOM/UFU

Servidor Web

- O servidor web (ou servidor HTTP) é um software no servidor capaz de tratar requisições HTTP (por ex., requisições do navegador solicitando arquivos HTML, CSS, imagens, etc.);
- O termo servidor web também é utilizado para referenciar o computador em si onde tal software está executando



Websites Estáticos x Dinâmicos

- Websites Estáticos
  - Conteúdo sempre o mesmo
  - HTML, CSS, JavaScript, imagens, etc.
  - Sem acesso a banco de dados
  - Servidor web estático (por ex. Apache HTTP)
- Websites Dinâmicos
  - Conteúdo produzido dinamicamente
  - Envolve programação **server-side**: PHP, Python, Java, etc.
  - Realizam processamentos adicionais no servidor como:
    - Acesso a arquivos e a banco de dados, processamento de formulários, gerencialmento de sessões, controle de login, etc.
  - Servidor web dinâmico (ex. Apache HTTP + PHP)

Aspectos Gerais sobre PHP

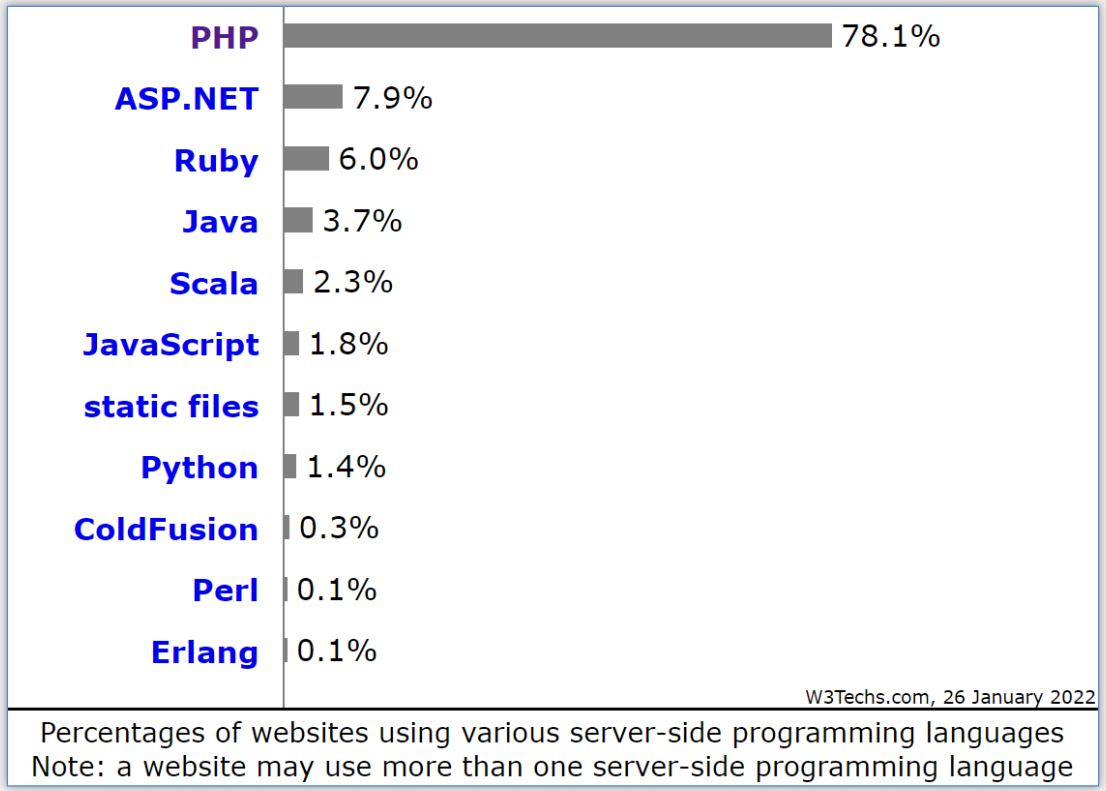
- PHP é um acrônimo recursivo para **P**HP: **H**ypertext **P**reprocessor
- É uma linguagem de script *server-side* que permite o desenvolvimento de websites dinâmicos
- Scripts em PHP são normalmente executados no servidor
- Operam em conjunto com o servidor HTTP

Aspectos Gerais sobre PHP

- Gratuito e Open Source
- Pode ser executado em várias plataformas
  - Windows, Linux, Unix, Mac OS
- Compatível com vários servidores HTTP
  - Apache HTTP, NGINX, Microsoft IIS, etc.
- Suporta vários sistemas de banco de dados
- Há vários frameworks: Laravel, Symfony, etc.
- Serviços gratuitos de hospedagem com suporte a PHP
- Muito popular e de fácil aprendizado

Linguagens Server-Side

Segundo o W3Techs, PHP é utilizada em 78% dos websites que se tem conhecimento da linguagem *server-side*



Fonte: W3Techs (Janeiro/2022)

Exemplos de websites populares que utilizam PHP

- facebook.com
- wikipedia.org
- pinterest.com
- yahoo.com
- mozilla.org
- wordpress.com

Algumas opções para começar com PHP

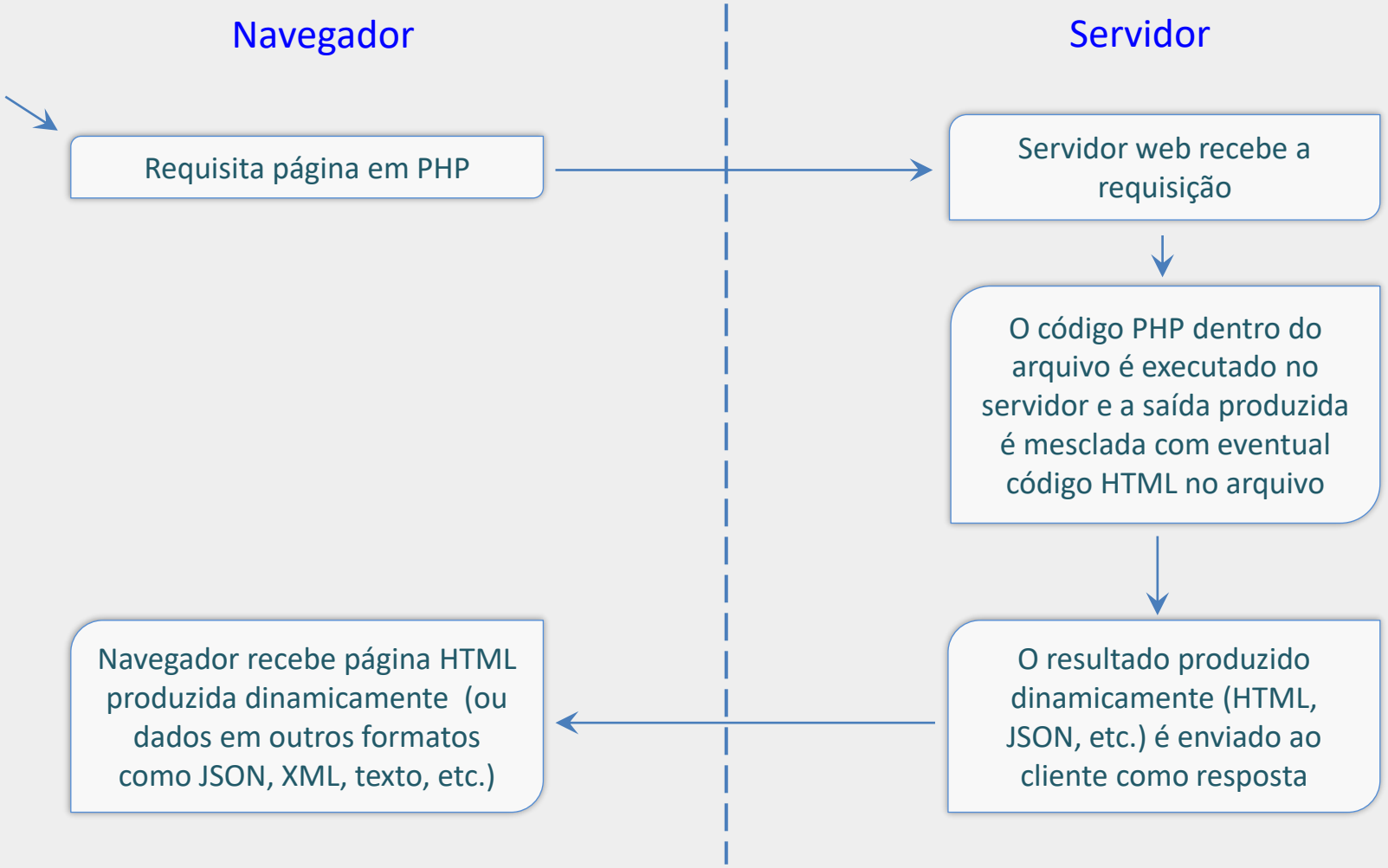
- Utilizar serviço de hospedagem com suporte a PHP
- Instalar e configurar manualmente o servidor web
  - Instalação de servidor HTTP (ex. Apache HTTP)
  - Instalação do PHP (php.net/downloads)
- Utilizar pacotes de instalação que incluem o PHP
  - Ex.: WAMP (**W**indows, **A**pache, **M**ySQL and **P**HP)

Hello World com PHP

Arquivo PHP

- Extensão .php
- Código PHP deve ser inserido dentro de <?php ?>
- Arquivo pode conter código PHP, HTML, CSS, etc.
- PHP é um pré-processador de hipertexto
  - Trechos de código PHP são pré-processados no servidor
  - Saída gerada pelo código PHP é mesclada com o restante
- Resultado final é enviado para o usuário (navegador)

Gerando Página Web Dinâmica com PHP



Hello World

Exemplo de script PHP

```php
<?php

   for ($i = 0; $i < 10; $i++) {
      echo "Hello World!";
   }

?>
```

Em PHP, utiliza-se bastante o construtor echo para produzir uma saída textual. Normalmente o conteúdo gerado com echo é enviado pelo servidor web como parte da resposta HTTP.

Hello World

```
<html>
<body>
  <h1>Página dinâmica com PHP</h1>

  <?php
    for ($i = 0; $i < 5; $i++) {
      echo "<p> Hello World! </p>";
    }
  ?>

</body>
</html>
```

```
<html>
<body>
 <h1>Página dinâmica com PHP</h1>

 <p> Hello World! </p>
 <p> Hello World! </p>
 <p> Hello World! </p>
 <p> Hello World! </p>
 <p> Hello World! </p>

</body>
</html>
```

Arquivo exemplo.php disponibilizado no servidor web (repare que o arquivo contém código HTML e PHP). Suponha que o arquivo tenha sido requisitado pelo navegador

Página HTML recebida pelo navegador (quando o arquivo exemplo.php é requisitado no servidor, o código PHP contido nesse arquivo será executado e a página HTML resultante é enviada ao navegador)

Hello World

```
<html>
<body>
 ...
 <h1>Página dinâmica com PHP</h1>
 <p> Parágrafo 1 </p>
 <?php
   echo "<p> Parágrafo 2 </p>";
 ?>
 <p> Parágrafo 3 </p>
 <?php
   echo "<p> Parágrafo 4 </p>";
 ?>
</body>
</html>
```

```
<html>
<body>
  ...
  <h1>Página dinâmica com PHP</h1>
  <p> Parágrafo 1 </p>
  <p> Parágrafo 2 </p>
  <p> Parágrafo 3 </p>
  <p> Parágrafo 4 </p>

</body>
</html>
```

É possível ter múltiplos trechos de código PHP no arquivo (exemplo3.php)

Saída produzida (página dinâmica)

PHP e Ajax - Exemplo

CEP

38408-100

Endereço

Av João Naves de Ávila

Bairro

Santa Mônica

Cidade

Uberlândia

Estado

Minas Gerais

- Os campos de endereço são preenchidos automaticamente assim que o usuário informa o CEP
- Um script PHP no servidor recebe o CEP como parâmetro e retorna um objeto JSON contendo os dados do endereço (que são recebidos pelo navegador e inseridos dinamicamente na página por script JS)

Recursos Básicos da Linguagem

Observações Gerais

- Declarações terminam com o ponto-e-vírgula
- Comentários de linha: `// comentário`
- Comentários de bloco: `/* comentário */`
- Sensível a letras maiúsculas e minúsculas
  - Nomes de variáveis e constantes
  - Chaves de arrays associativos
- Não sensível a maiúsculas e minúsculas
  - Palavras reservadas da linguagem
  - Nomes de classes, métodos e funções

Operadores

Operador	Significado
+	Adição (e concatenação)
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão inteira
++	Incremento
--	Decremento
=	Atribuição
+=	Atribuição com soma
-=	Atribuição com sub.
.	Concatenação
**	Exponenciação

Operador	Significado
==	Comparação por igualdade
===	Comparação por igualdade, incluindo valor e tipo
!= ou <>	Diferente
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
&& ou and	"E" lógico
|| ou or	"Ou" lógico
!	Negação lógica

Estruturas Condicionais e de Repetição

```php
if (expressão) {
   // operações se verdadeiro
}
else {
   // operações se falso
}
```

```php
switch (expressao) {
    case condicao1:
        // operações
        break;

    case condicaoN:
        // operações
        break;
    ...
    default:
        // operações
}
```

```php
for ($i = 0; $i < 10; $i++)
{
   // operações
}
```

```php
foreach ($array as $elem)
{
   // operações
}
```

```php
while (expressao)
{
   // operações
}
```

```php
do {
   // operações
} while (expressao)
```

Variáveis

- Começam com o símbolo $
- São declaradas na primeira atribuição
- O tipo é definido automaticamente

```php
$x = 5;                 // variavel do tipo integer
$y = -6;                // variavel do tipo integer
$z = 3.14;              // variavel do tipo float
$str = "bla bla";       // variavel do tipo string
$cond = true;           // variável booleana
$pares = [2, 4, 6];     // array
```

Constantes

- Definidas por meio da função define
- Possuem escopo global
- Letras maiúsculas são utilizadas (por convensão)

```php
<?php
    define('PATH_UPLOADS', '/arqs_uploads');
    define('DB_NAME', 'mysqlTeste');
    define('DB_PSWD', '123456');
?>
```

OBS: Dentro de classes: ***const*** *PI = 3.14;*

Strings

- Podem ser definidas com aspas duplas
  - Conteúdo da string é avaliado
  - Pode conter nomes de variáveis
- Podem ser definidas com aspas simples
  - String não avaliada
  - Apenas texto
- Podem ser definidas com a sintaxe Heredoc
  - Texto delimitador no início e no fim
  - Strings longas, múltiplas linhas, sem aspas
  - Conteúdo da string é avaliado

Strings e o construtor **echo**

```php
$idade = 15;
$mes = 10;
$dist = 30;

echo "A idade eh $idade";  // a saída será: A idade eh 15
echo 'A idade eh $idade';  // a saída será: A idade eh $idade

// Para imprimir uma variável sem espaço/caracter especial depois
// so seu nome, coloque o nome da variável entre chaves
echo "Distância de {$dist}km";   // Distância de 30km
```

String **Heredoc**

```php
$str = <<<STRING_HEREDOC

Texto com múltiplas linhas.
Aqui dentro posso utilizar aspas simples '
e também aspas duplas "
assim como nomes de variáveis como $exemplo

STRING_HEREDOC;
```

- STRING_HEREDOC é apenas um identificador. Porderia ser, por exemplo, <<<SQL ou <<<HTML
- Não deve haver espaços depois de <<<STRING_HEREDOC

String **Heredoc** - Exemplo

Exemplo

```
$maxSal = 10;
$str = <<<SQL
    SELECT *
    FROM Funcionario
    WHERE salario > $maxSal and
          codDepart = 'Vendas'
    SQL;

echo $str;
```

Saída produzida

```
SELECT *
FROM Funcionario
WHERE salario > 10 and
      codDepart = 'Vendas'
```

- O identificador de fechamento só pode ser indentado a partir do PHP 7.3
- O espaçamento utilizado na linha do delimitador de fechamento (antes de SQL; no exemplo acima) será removido de cada linha da string. Se não for possível remover, ocorrerá um erro (veja o próximo slide)

String **Heredoc** - Exemplo

```
$maxSal = 10;
$str = <<<SQL
  SELECT *
    FROM Funcionario
    WHERE salario > $maxSal and
          codDepart = 'Vendas'
    SQL;
```

Erro de sintaxe. *Primeira linha da string não começa com a quantidade mínima de espaços, conforme delimitador de fechamento.*

Funções

Função simples

```
function max($a, $b)
{
    if ($a > $b)
        return $a;
    else
        return $b;
}
```

Parâmetros com tipo definidos

```
function max(float $a, float $b)
{
    if ($a > $b)
        return $a;
    else
        return $b;
}
```

Parâmetros por referência

```
function atualizaVar(&$a)
{
    $a = $a * 1.2;
}
```

Chamada da função

```
$maior = max(2, 5);
$a = 10;
atualizaVar($a);
```

Variáveis Globais

```php
<?php

    $x = 10;    // $x é global

    function exemplo() {
        echo $x;     // erro
        global $x;
        echo $x;     // 10
    }

?>
```

Necessário utilizar `global` para acessar variável global dentro de funções

Array Indexado

```php
$pares1 = [2, 4, 6];  // Definindo array com colchetes
$pares2 = array(2, 4, 6); // Definindo array com construtor array

// Definindo array vazio e adicionando elementos no final
$pares = [];
for ($i = 0; $i < $n; $i++) {
  echo $pares[] = $i;
};

// Percorrendo array com estrutura for simples
$n = count($pares);
for ($i = 0; $i < $n; $i++) {
  echo $pares[$i];
};

// Percorrendo array com estrutura foreach
foreach ($pares as $par) {
  echo $par;
};
```

Array Associativo

```php
$alunos = array(
    "GSI010" => "Augusto",
    "GSI011" => "Camila",
    "GSI012" => "Pedro"
);

echo $alunos['GSI011'];   // a saída será 'Camila'
echo $alunos[0];          // ocorrerá um erro!
```

- Cada elemento é acessado por meio da chave (*key*)
- A chave pode ser *string* ou *inteiro*
- O valor pode ser de qualquer tipo

Arrays **Super Globais**

Arrays **super globais** são arrays associativos especiais que podem ser acessados de qualquer lugar

Array	Descrição
$GLOBALS	Permite acessar as variáveis globais do script
$_GET	Acesso a campos de formulários submetidos com GET
$_POST	Acesso a campos de formulários submetidos com POST
$_FILES	Acesso a arquivos anexados a formulários
$_COOKIE	Acesso a cookies enviados ao servidor
$_SESSION	Acesso a variáveis de sessão
$_SERVER	Informações sobre o servidor, navegador, etc.

Arrays **Super Globais**

Array	Descrição
`$_SERVER['PHP_SELF']`	Nome do arquivo do script que está em execução
`$_SERVER['REQUEST_METHOD']`	Método da requisição HTTP utilizado para acessar a página
`$_SERVER['HTTP_USER_AGENT']`	Contém informações sobre o navegador
`$_SERVER['REMOTE_ADDR']`	Endereço IP do usuário que está acessando a página

Classes e Objetos

```php
class Circulo
{
    // Declaração de atributos.
    // O modificador de acesso padrão
    // é public (caso seja omitido)
    private $raio = 0;
    private $area = 0;

    // Construtor
    function __construct($raio)
    {
        $this->raio = $raio;
        $this->area = 3.14*$raio*$raio;
    }

    // Declaração de método
    public function mostraArea()
    {
        echo $this->area;
    }
}
```

```php
<?php

$circ1 = new Circulo(5);
$circ1->mostraArea();

?>
```

Processamento de Formulários

Recebendo Formulários - Método POST

Formulário HTML - **Método POST**

```
<form action="processaForm.php" method="POST">
    Nome: <input type="text" name="nome" >
    Email: <input type="email" name="email">
</form>
```

```
<?php
    $nome = $_POST["nome"];
    $email = $_POST["email"];
    echo "Sr. $nome, seu e-mail é $email";
?>
```

Arquivo processaForm.php

Atenção: Utilize o atributo **name** para fornecer um nome de identificação para o campo.

Posteriormente, utilize essa identificação para recuperar o dado no PHP.

Um erro comum é utilizar o atributo id para esse propósito.

Recebendo Formulários - Método GET

Formulário HTML - **Método GET**

```
<form action="processaForm.php" method="GET">
    Nome: <input type="text" name="nome">
    Email: <input type="email" name="email">
</form>
```

```
<?php
    $nome = $_GET["nome"];
    $email = $_GET["email"];
    echo "Sr. $nome, seu e-mail é $email";
?>
```

Arquivo processaForm.php

Formulários com Campos Vazios

```php
<?php
  $nome = isset($_POST["nome"]) ? $_POST["nome"] : "";
  $email = isset($_POST["email"]) ? $_POST["email"] : "";
  echo "Sr. $nome, seu e-mail é $email";
?>
```

Utilizando a função **isset** em conjunto com operador **?**

Para prevenir eventuais erros causados por campos vazios, pode-se utilizar a função **isset** do PHP

```php
<?php
  $nome = $_POST["nome"] ?? "";
  $email = $_POST["email"] ?? "";
  echo "Sr. $nome, seu e-mail é $email";
?>
```

Forma equivalente utilizando operador **??**

Uma alternativa mais prática é utilizar o operador ??, que permite resgatar o valor da variável apenas caso ela esteja definida

Exemplo Simples de Validação

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

    // Não deixe de validar os campos
    if (trim($nome) == "")
        $errorMsg = "O nome é obrigatório";
?>
```

A validação dos campos no lado servidor é recomendada mesmo que eles já tenham sido validados no lado cliente.
A função **trim** retorna uma nova string após remover espaços do início e do fim da string passada como parâmetro (assim como tabulações e quebras de linhas)

Cuidado com Ataques **XSS**!

```
<html>
<body>
    <h1>Site Vulnerável à Ataques XSS</h1>
    <?php
        $nome = $_GET["nome"];
        $email = $_GET["email"];
        echo <<<HTML
        <table>
            <tr>
                <td> $nome </td>
                <td> $email </td>
            </tr>
        </table>
        HTML;
    ?>
</body>
</html>
```

Cross-Site Scripting (XSS)
Ataque malicioso onde um código potencialmente prejudicial é injetado utilizando a URL ou campos de formulário

Cuidado com Ataques **XSS**!

```
<html>
<body>
    <h1>Prevenindo XSS</h1>
    <?php
        $nome = $_GET["nome"];
        $email = $_GET["email"];
        ...
        $nome = htmlspecialchars($nome);
        $email = htmlspecialchars($email);
        echo <<<HTML
        <table>
            <tr>
                <td> $nome </td>
                <td> $email </td>
            </tr>
        </table>
        HTML;
    ?>
...
```

htmlspecialchars
Pode ser utilizada para "sanitizar" um conteúdo produzido pelo usuário, ***antes de inserir*** *esse conteúdo na página HTML*

htmlspecialchars *converte, por padrão:*

- *< em <*
- *> em >*
- *& em &*
- *" em "*

Mas Atenção: *não é uma boa prática armazenar o conteúdo (no banco de dados, por exemplo) após passar pela função* ***htmlspecialchars***

Recebendo Parâmetros pela URL

Exemplo de link gerado dinamicamente
com passagem de parâmetros pela URL

```
<a href="detalhes.php?codProd=1"> Notebook </a>
<a href="detalhes.php?codProd=2"> Celular </a>
```

```
<?php
    $codigoProduto = $_GET["codProd"];
    // busca pelo produto
?>
```

Resgate do parâmetro no arquivo detalhes.php

Manipulando Senhas

```php
<?php

// o hash gerado terá 60 caracteres, mas pode aumentar
$senhaHash = password_hash($senha, PASSWORD_DEFAULT);

// gravar no BD o hash da senha e não a senha
gravaNoBD($senhaHash);

?>
```

Cadastro do Usuário

*Uso da função **password_hash** do PHP para gerar um hash de senha seguro utilizando o algoritmo **BCrypt** antes de grav no BD*

```php
<?php

// resgatar do BD a senha hash do usuário
$senhaHash = resgataSenhaHashDoBD();

if (password_verify($senhaFornecida, $senhaHash) {
    // Login efetuado com sucesso!
}

?>
```

Validação de Login

*Uso da função **password_verify** para comparar a senha fornecida com o hash da senha armazenado.*

Redirecionando com PHP

```php
<?php
    // pagina.php
    header("Location: nova-pagina.php");
    exit();
?>
```

- **header** envia um cabeçalho HTTP
- Deve ser chamada antes do script produzir qualquer saída
  - Antes de código HTML, antes de ***echo***, etc.
- É uma boa prática chamar **exit**() depois do redirecionamento para garantir que nenhum conteúdo seja produzido a partir desse ponto

Uso Inadequado da Função header

```
<html>
<body>
  <h1>Ex. de uso INADEQUADO da função header</h1>

  <?php
    header("Location: nova-pagina.php");
    exit();
  ?>

</body>
</html>
```

Não faça isso!

Neste exemplo a função header não terá efeito, pois o arquivo PHP produz uma saída HTML antes de sua chamada (por causa do código HTML no início do arquivo)

include e require

- **include** e **require** incluem outro arquivo PHP no script
- O arquivo incluído pode conter variáveis, funções, classes, etc.
- **include** gera um *warning* em caso de falha na inclusão
- **require** gera um *erro fatal* e encerra o script
- Não são funções!

```php
<?php
    // mysqlConnection.php pode conter
    // dados de conexão com o MySQL,
    // por exemplo
    include "mysqlConnection.php";
?>
```

include_once e require_once

- include_once e require_once são similares a include e require
- Porém, não incluem o arquivo caso ele já tenha sido incluído
- include e require produzem warnings/erros nessa situação

Referências

- https://www.php.net/docs.php
- NIXON, R. ***Learning PHP, MySQL & JavaScript:*** With jQuery, CSS & HTML5. 5. ed. O'Reilly Media, 2018