

Técnico em desenvolvimento de sistemas
SENAI SUL

FIESC SENAI

sc.senai.br

Programação web

Professor Rogério dos Santos

email:rogerio.santos@edu.sc.senai.br

Pós graduado com especialização em linguagem java

O que é PHP ?

PHP é uma linguagem que permite criar sites WEB dinâmicos, possibilitando uma interação com o usuário através de formulários, parâmetros da URL e links.

O que é um servidor web ?

É um aplicativo que recebe a requisição de página e responde com o código fonte da página selecionada.

O browser recebe o código e solicita imagens e demais arquivos necessários para correta exibição da página. Isso é chamado de código client side, pois é interpretado pelo computador do usuário.

E onde entra o PHP ?

- ✓ É executado e renderizado no servidor web.
- ✓ É necessário entender: PHP é uma linguagem de servidor, o internauta recebe somente HTML.
- ✓ É correto dizer que o meu site é feito em PHP, o que é muito melhor que HTML ?

O que pode ser feito com PHP ?

- Coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber *cookies*.
- PHP também tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como dBase, Interbase, mSQL, mySQL, Oracle, Sybase, PostgreSQL e vários outros.

- Coletar dados de um formulário, gerar páginas dinamicamente ou enviar e receber *cookies*.
- PHP também tem como uma das características mais importantes o suporte a um grande número de bancos de dados, como dBase, Interbase, mSQL, MySQL, Oracle, Sybase, PostgreSQL e vários outros.

Construir uma página baseada em um banco de dados torna-se uma tarefa extremamente simples com PHP.

SURGIMENTO DA LINGUAGEM WEB

A linguagem PHP foi concebida durante o outono de 1994 por **Rasmus Lerdorf**. As **primeiras versões** não foram disponibilizadas, tendo sido utilizadas em sua *home-page apenas para que ele pudesse ter informações sobre as visitas* que estavam sendo feitas.

- A primeira versão utilizada por outras pessoas foi disponibilizada em 1995, e ficou conhecida como **“Personal Home Page Tools” (ferramentas para página pessoal)**.
- Era composta por um sistema bastante simples que interpretava algumas *macros e alguns utilitários que rodavam “por trás” das home-pages: um livro de visitas, um contador e algumas outras coisas.*

Em meados de 1995 o interpretador foi reescrito, e ganhou o nome de **PHP/Fl**, o “**Fl**” veio de um outro pacote escrito por Rasmus que interpretava dados de formulários HTML (**Form Interpreter**). Ele combinou os scripts do pacote Personal Home Page Tools com o Fl e adicionou suporte a MySQL, nascendo assim o PHP/Fl, que cresceu bastante, e as pessoas passaram a contribuir com o projeto.

Características da linguagem

- Não tem custo de licença;
- Total integração com HTML, JavaScript e outras tecnologias derivadas;
- Totalmente dinâmica;
- Permite o acesso aos arquivos de texto, imagens e outros tipos de arquivos;
- Privilegia a segurança de dados a partir da criptografia de dados.

- Suporte a banco de dados variados com acesso nativo: Firebird, Interbase,MySQL SQLSERVER,Oracle,PostgreSql entre outros;
- Roda no lado do servidor o que garante integridade do seu script.

SINTAXE BÁSICA

```
<?php  
    comandos  
?>
```

Exemplo:

```
<?php  
    echo "Testando a sintaxe geral do PHP"  
?>
```

Um outro exemplo:

```
<?
```

```
    print("Testando a sintaxe geral do PHP")
```

```
?>
```

A notação acima é mais usada para uso a partir da versão 5

Pode-se trabalhar com outras tags para delimitar o início e o fim da tag.

Exemplo:

<script language="php">

VARIÁVEIS

REGRAS PARA NOME DE VARIÁVEIS EM PHP.

- ✓ Toda variável em PHP começa com o caracter \$ seguido pelo nome da mesma.Ex: \$Total;
- ✓ Os nomes das variáveis em PHP fazem distinção entre maiúscula e minúscula.
Ex:\$idade não é igual a \$Idade;

- Um nome válido de variável deve começar com caractere _ ou uma letra, seguido de qualquer quantidade de letras, algarismos ou _. Ex: \$_texto, \$total, SALARIO são nomes válidos.
- \$7cidades, \$*multiplica são nomes inválidos

Para usar variáveis não se determina o tipo, pois a linguagem PHP não é fortemente tipada como Delphi, Java e outras linguagens, isto significa que a variável assume o tipo de dado que lhe for atribuído.

EXEMPLOS:

\$LOGICO2 = FALSE

É atribuído o valor lógico FALSE na variável \$LOGICO2, é tido como booleano, nesse caso a variável passa a ser booleana.

\$INTEIRO=49

É atribuído a variável \$INTEIRO o valor 49, isto é tido como número inteiro, a variável passa a ser numérica do tipo inteira.

\$PFLUTUANTE=10.30

É atribuído a variável \$PFLUTUANTE o valor de 10.30, isto é tido como um número de ponto flutuante ou com casas decimais, então a variável passa a ser do tipo flutuante.

\$CARACTER="TIPO STRING"

É atribuído a variável \$CARACTER o valor de "TIPO STRING", isto é tido como uma cadeia de caracteres ou string.

```
$arvalor=array("nome"=>"sobrenome"=>"silva")
```

É atribuído à variável \$arvalor um array contendo

o nome e sobrenome de uma pessoa, então a variável passa a ser do tipo array[].

Atribuição de valores

Em relação aos exemplos anteriores, valores foram atribuídos nas variáveis na seguinte sintaxe:

\$variavel = valor

É uma atribuição direta de valores onde na esquerda da sentença tem a variável que receberá o valor, seguida do símbolo de “=” onde é o operador que determina atribuição, seguido do valor que será atribuído na variável

Tipos suportados

TIPO: BOOLEAN

FAIXA DE VALORES: (True) verdadeiro) ou (False)

APLICAÇÃO: Esse tipo deve ser utilizado para fazer validações lógicas. Ex: \$flag = true.

TIPO: INTEGER

FAIXA DE VALORES: -2.147.483.648 A 2.147.483.647

APLICAÇÃO: Esse tipo de variável pode ser utilizado para representar a idade de uma pessoa, mas não para fazer cálculos que necessitam de precisão decimal.

TIPO: FLOAT

FAIXA DE VALORES: 1.40 E-45 a 3.40 E+38

APLICAÇÃO: É um tipo utilizado quando precisa armazenar número que necessita de precisão decimal. Não se utiliza esse tipo para fazer comparações de igualdades pois isso pode gerar erros difíceis de serem identificados principalmente se o valores comparados tiverem muitas casas decimais.

Ex: \$valor=45.50

TIPO:STRING

FAIXA DE VALORES: NÃO TEM

APLICAÇÃO: Esse tipo é geralmente utilizado para se trabalhar com caracteres em geral.

Ex: \$cidade = "JOINVILLE".

TIPO: ARRAY

FAIXA DE VALORES: NÃO TEM

APLICAÇÃO: Serve para representar uma coleção de tipo heterogêneo.

Ex: \$valores=array("requirido"=true,1=>20)

TIPO: OBJECT

FAIXA DE VALORES: NÃO TEM

APLICAÇÃO: O tipo nada mais é do que uma instância de uma classe. Ex: \$pessoa = new pessoa()

TIPO: NULL

FAIXA DE VALORES: NÃO TEM

APLICAÇÃO: É um valor especial que representa uma variável que não possui um valor definido.

- Um fato importante no uso de variáveis é que como no PHP não é necessário declarar a variável para depois atribuir, uma variável que pode ser atribuída com valores do tipo caracter, também pode ser atribuída com valores do tipo inteiro.
- Ex: `$teste = "Cadeia de caracteres";`
- `$teste = 10;`

Declaração de constantes

São valores que são predefinidos no início do programa, e que não mudam ao longo de sua execução. Pode-se definir suas próprias constantes utilizando o comando DEFINE.

Exemplo:

```
Define("Disciplina", "Programação Web");
```

```
Define("Nota",8);
```

```
Echo "A Disciplina É". Disciplina;
```


```
Echo "A nota é" .Nota;
```

Na declaração da constante o nome da constante pode ser definido se pode ser usado com caixa alta ou não. Exemplo:

```
Define("Disciplina", "Programação Web",true);
```

```
Echo "A disciplina é ".Disciplina;
```

```
Echo "A disciplina é ".disciplina;
```



Quando o parâmetro é **true** o nome da constante pode ser usado caixa alta ou baixa. Quando o omissso ou false o nome da constante pode ser usado conforme o que foi definido.

Linhas de comentário

No php as linhas de comentário segue a seguinte sintaxe:

✓ **Comentário de uma linha:**

```
/*Comentário
```

✓ **Comentário de mais de uma linha:**

```
/* Este é um comentário de mais de uma linha  
na linguagem PHP */
```

Troca de informações

- ✓ Conceitos sobre troca de informações
- ✓ Enviando informações via GET
- ✓ Enviando informações via POST

✓ Nesse contexto temos os métodos GET e POST, que são necessários para que a página PHP receba informações.

✓ E para que serve isso ?

Você já deve ter preenchido um formulário via WEB, neste caso você está enviando uma informação.

✓ A página PHP pode receber essas informações tanto por GET como por POST.

✓ Em linhas gerais:

GET: Quando é enviado informações pela URL;

POST: Quando é enviado informações através dos campos de formulário.

Operadores Aritméticos

Operador	Função
+	Somar valores. Ex: \$valor1 + \$valor2
-	Subtrair valores. Ex: \$valor1 - \$valor2
*	Multiplicar valores. Ex: \$valor1 * \$valor2
/	Dividir valores. Ex: \$valor1 / \$valor2
%	Módulo de valores: resto da divisão inteira entre os números. Ex: \$valor1 % \$valor2

Fonte: Melo e Nascimento (2007, p. 57)

Operadores de comparação e atribuição.

COMPARAÇÃO : São aqueles utilizados em estruturas de controle, do tipo, caso, enquanto faça, para-faça, ou seja, sempre que houver a necessidade de se fazer uma comparação entre 2 ou mais valores.

ATRIBUIÇÃO: São operadores que servem para atribuir valores às variáveis.

Operador	Função
=	Operador de atribuição. Ex: \$valor1=1
==	Operado de igualdade, retorna <i>true</i> se dois valores são iguais. Ex:\$valor1 ==\$valor2
===	Operador identidade de valor e tipo retorna <i>true</i> se dois valores são iguais e do mesmo tipo. Ex:\$valor1 === \$valor2
!= ou <>	Operador de diferença retorna <i>true</i> se os dois valores são diferentes. Ex: \$valor1 != \$valor2 ou \$valor1 <> \$valor2
!==	Operador de não identidade de valor e tipo. Ex: \$valor1 !== \$valor2
<	Menor que. Ex: \$valor1 < \$valor2
>	Maior que. Ex:\$valor1 > \$valor2
<=	Menor ou igual que. Ex: \$valor1 <= \$valor2
>=	Maior ou igual que. Ex: \$valor1 >= \$valor2

Operadores de incremento/decremento e aritméticos com atribuição.

Operador	Função
<code>++\$var</code>	Incrementa \$var e então retorna \$var incrementada.
<code>--\$var</code>	Decrementa \$var e então retorna \$var decrementada.
<code>\$var++</code>	Retorna \$var e depois a incrementa.
<code>\$var--</code>	Retorna \$var e depois a decrementa.
<code>+=</code>	Soma o conteúdo de uma variável com um valor, e depois reatribui o resultado nela própria. Ex: <code>\$var += 5;</code> equivale a codificar <code>\$var = \$var + 5;</code>
<code>-=</code>	Subtrai o conteúdo de uma variável com um valor, e depois reatribui o resultado nela própria. Ex: <code>\$var -= 5;</code> equivale a codificar <code>\$var = \$var - 5;</code>
<code>*=</code>	Multiplica o conteúdo de uma variável com um valor, e depois reatribui o resultado nela própria. Ex: <code>\$var *= 5;</code> equivale a codificar <code>\$var = \$var * 5;</code>
<code>/=</code>	Divide o conteúdo de uma variável com um valor, e depois reatribui o resultado nela própria. Ex: <code>\$var /= 5;</code> equivale a codificar <code>\$var = \$var / 5;</code>

Fonte: Melo e Nascimento (2007, p. 59)

Operadores Lógicos

Operador	Nome	Função
<i>and</i> e &&	E	Retorna verdadeiro quando os dois valores avaliados são verdadeiros. Caso contrário retorna falso. A diferença entre usar <i>and</i> ou && é a precedência dos mesmos.
<i>or</i> e	OU	Retorna verdadeiro quando pelo menos um dos dois valores avaliados é verdadeiro. Caso contrário retorna falso. A diferença entre usar <i>or</i> ou é a precedência dos mesmos.
Xor	Ou Exclusivo	Retorna verdadeiro se um, e apenas um dos dois valores, for verdadeiro. Caso contrário, retorna falso.
!	Não	Retorna verdadeiro se o valor avaliado for falso. Caso contrário retorna verdadeiro.

Fonte: Melo e Nascimento (2007, p. 60)

Estrutura de controle

ESTRUTURA IF

Essa estrutura verifica se uma condição é verdadeira ou não. Se for verdadeira é executada as instruções dentro do bloco, caso contrário é desviado o fluxo de instrução.


```
<html>
<body>
  <?php
    $valor = 90;
    if($valor == 10)
      echo "O valor e igual a 10";
    else
      echo "O valor nao e igual a 10";
  ?>
</body>
</html>
```

ESTRUTURA IF

Um outro exemplo com IF é se caso uma instrução seja verdadeira e queira que seja executado mais de um instrução que no caso é usado { para abertura de bloco e } para fechamento do bloco

```
<html>
<body>
  <?php
    $valor = 10;
    if($valor == 10)
    {
      echo "O valor e igual a 10". "<br>";
      $total = $valor + 5;
      echo " O total e ".$total;
    }
    else
    {
      echo "O valor nao e igual a 10". "<br>";
      echo "Nao pode fazer o calculo";
    }
  ?>
</body>
</html>
```

ESTRUTURA IF ELSEIF

A linguagem PHP permite a instrução if elseif que na tradução é SE SENAO SE, dependendo da necessidade lógica. Um exemplo vamos ver a seguir.

```
<html>
<body>
  <?php
    $valor = 1;
    if($valor == 10)
    {
      echo "O valor e igual a 10". "<br>";
      $total = $valor + 5;
      echo " O total e ".$total;
    }
    elseif ($valor >=10)
    {
      echo "O valor e maior que 10". "<br>";
      $total1 = $valor * 3;
      echo "Resultado ".$total1;
    }
    else
    {
      echo "O valor nao e igual a 10 e nem maior". "<br>";
      echo "Nao pode fazer o calculo";
    }
  ?>
</body>
</html>
```

Aninhamento de estrutura IF

Dependendo da condição lógica pode ser necessário usar um IF dentro do outro IF.

Isto é chamado de aninhamento de IF

Um exemplo é mostrado a seguir.

Estrutura de controle – IF ELSE

```
If($condicao1)
{
    //$condicao1 verdadeira
    If($condicao2)
    {
        //$apenas executa se $condicao1 e $condicao2 forem verdadeiras
    }
    }else if($condicao3)
    {
        //$executa se $condicao1 for falsa e $condicao3 verdadeira
    }else {
        //$executa se $condicao1 e $condicao3 forem falsas
    }
}
```

Estrutura de controle - SWITCH

Essa estrutura, a exemplo do if, também é uma estrutura do tipo condicional, podendo ser muito útil e eficiente quando se deparar de trabalhar com estruturas condicionais aninhadas.


```
<html>
<body>
  <?php
    $condicao = "B";
    switch ($condicao)
    {
      case "A":
        echo " letra a";
        break;

      case "B":
        echo " letra B";
        break;

      case "C":
        echo " letra C";
        break;
      default:
        echo "Nao e nenhuma dessas letras";
        break;
    }
  ?>
</body>
</html>
```

Estrutura de repetição

WHILE()

Uma estrutura de repetição serve para que possa executar uma ou mais linhas de instruções até que certa condição seja verdadeira.

Estrutura While

```
<html>
<body>
  <?php
    $contadora=1;
    //Atribuição de um valor inicial para a variável controladora do loop
    while($contadora<=5) //Início da estrutura com a condição lógica
    { //Início do bloco de instruções
      //Instruções executadas caso a condição seja verdadeira
      echo $contadora. " - " . "ESTRUTURA DE REPETICAO DO TIPO WHILE( )". "<br>";
      $contadora++; //Somar 1 na variável, a cada execução da estrutura
    } //Término do bloco de instruções
  ?>
</body>
</html>
```

ESTRUTURA DE REPETIÇÃO FOR

Essa estrutura de repetição é usada quando se determina quantas vezes as linhas de instruções serão executadas.

```
<html>
<body>
  <?php
    $contadora=1;
    for ($contadora = 1;$contadora <=5; $contadora++)
      echo $contadora. " - ". "ESTRUTURA DE REPETICAO FOR( )". "<br>";
    .....
  ?>
</body>
</html>
```

Um outro exemplo com FOR é quando se tem várias linhas que serão processadas então as mesmas serão executadas dentro de um bloco de {}. O exemplo será mostrado a seguir.

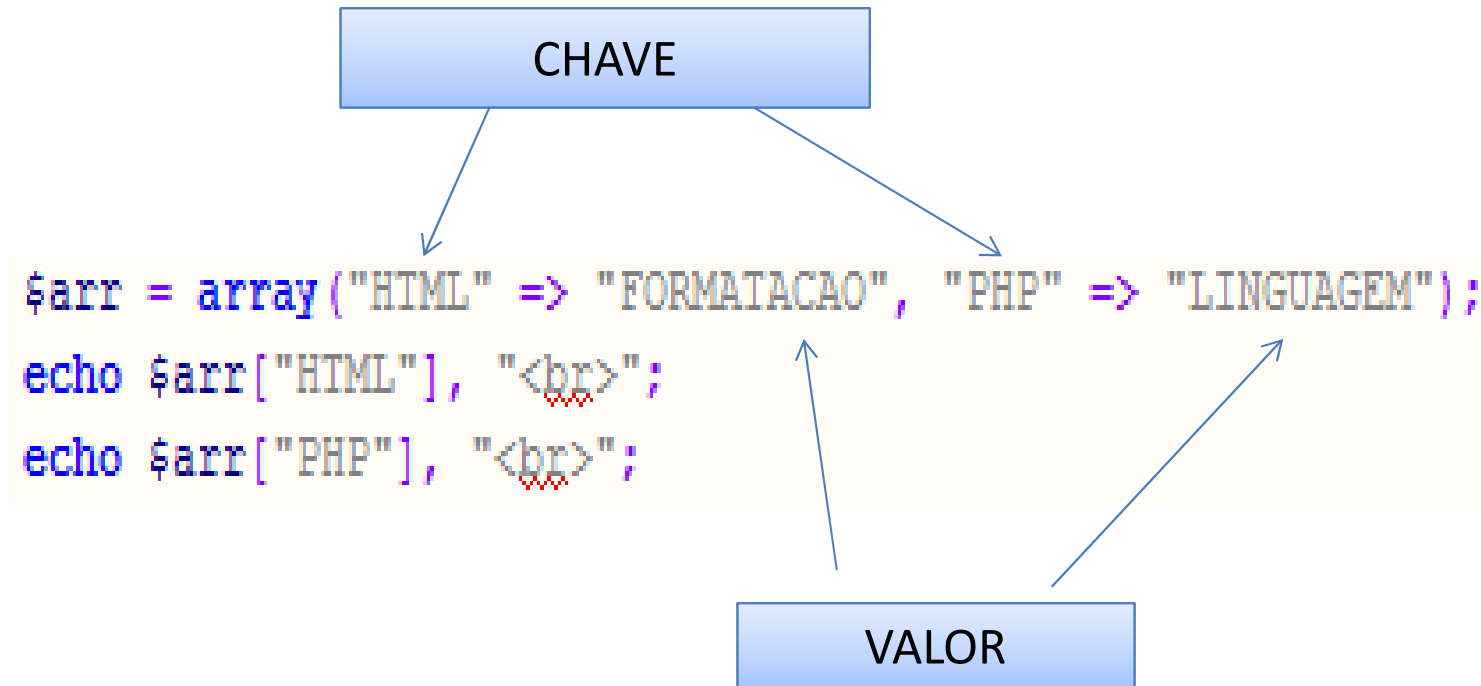
```
<html>
<body>
  <?php
    $contadora=1;
    for ($contadora = 1;$contadora <=5; $contadora++)
    {
      echo $contadora. " - " . "ESTRUTURA DE REPETICAO FOR( )". "<br>";
      $valor = $contadora + 1;
    }
    echo "Valor = ".$valor;
  ?>
</body>
</html>
```

Array

ARRAY SIMPLES

- No PHP arrays são mapas ordenados de chaves e valores, ou seja, pode-se atribuir a um elemento do array uma chave e um valor.
- Dessa forma é possível representar listas, coleções, dicionários, pilhas,filhas, etc.
- Existem duas formas definir um array no PHP.
`Array([chave1] => valor, [chave2} =>valor..)`

Exemplo com Array



RESULTADO



FORMATAÇÃO
LINGUAGEM

```
$arr1 = array("HTML", "PHP" => "LINGUAGEM");  
echo $arr1[0]. "<BR>";  
echo $arr1["PHP"];
```

RESULTADO



HTML

LINGUAGEM

```
$arr1 = array("HTML", "PHP" => "LINGUAGEM");  
print_r($arr1);
```

A função `print_r` imprime o array mostrando a posição do índice e o valor

RESULTADO



Array ([0] => HTML [PHP] => LINGUAGEM)

- Uma outra forma de definir um array é a forma direta, ou seja, apenas atribuímos a um elemento do array, um valor conforme a sintaxe abaixo
- `$arr[chave]= valor;`

```
$a1["Cliente 1"] = 1000;
```

```
$a1["Cliente 2"] = 5000;
```

```
$a1["Cliente 3"] = 0;
```

```
print_r($a1)."<br>";
```

```
$b1 = array("Cliente 1" =>100, "Cliente 2" =>5000, "Cliente 3 "=> 0);
```

```
print_r($b1);
```


Funções

- Criar funções em um programa são úteis para deixar o código dos programas mais organizados;
- Programas com funções deixa os programas modulares;
- Poupam tarefas de repetir determinado código toda vez que é preciso realizar a mesma tarefa;

Sintaxe de uma função com passagem de parâmetro por valor

```
FUNCTION nome_funcao(arg1,arg2,arg3...argn)  
{  
    comandos  
    [return <expressao>]  
}
```

Nome da função

\$valor1, \$valor2 e \$valor3 são parâmetros por valor

```
<?php
function soma_valores($valor1,$valor2,$valor3)
{
    $soma = $valor1 + $valor2 + $valor3;
    echo "A soma dos valores e ". $soma;
}

$sn1 = 10;
$sn2 = 20;
$sn3 = 30;
soma_valores($sn1,$sn2,$sn3);
?>
```

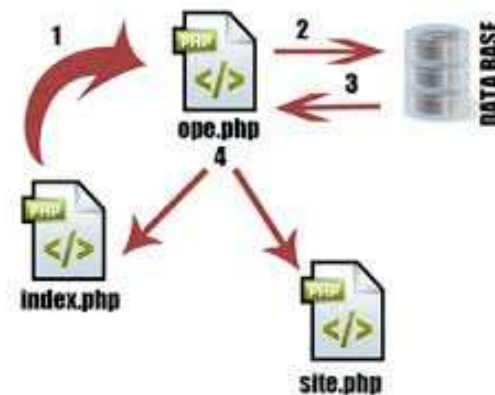
Exemplo de função com passagem de parâmetro por referência.

```
<?php
/*Uso de função com passagem de Parâmetros por referência*/
//Função dobra
function dobra(&$valor)
{
    $valor *=2;
}
//Script principal
$numero = 10;
dobra($numero);
echo "Resultado calculado pela funcao dobra e : ".$numero;
?>
```

Na passagem de valor por referência a variável \$valor é antecedita pelo caractere &, isto significa que esta conterá um ponteiro para a variável \$número, localizada no *script principal* da aplicação. Uma curiosidade neste processo, qualquer modificação feita em \$valor, automaticamente, é refletida em \$número, o que explica o resultado obtido no exemplo anterior.

Sessões

Sessões em php constituem um meio de guardar dados de uma forma que seja possível acessar estes em entradas subsequentes, além do fato de serem uma ótima alternativa de armazenamento de dados.

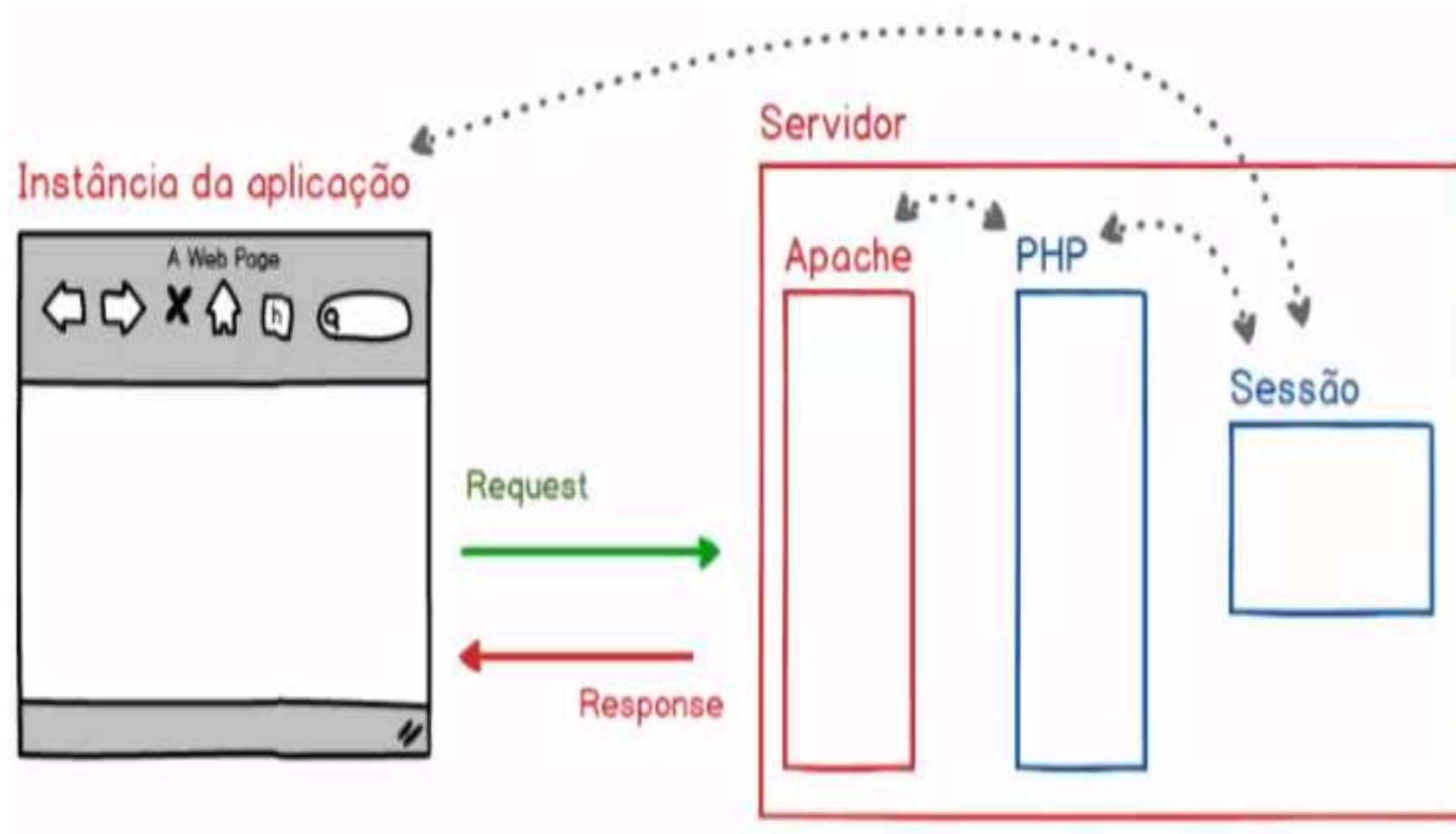


Como funciona

- ✓ A partir do momento que o usuário acessa a o site na internet uma sessão é criada entre o usuário e o servidor.



Como funciona



- ✓ Nada mais é que um **espaço de memória** onde é possível do lado do **servidor armazenar algumas informações** que conecte a instância da aplicação do **lado do servidor** com a instância da aplicação do **lado do cliente** criando uma ponte.



✓ **SESSION_START()**

Cria uma sessão ou restaura os dados de uma sessão atual, com base no identificador fornecido(passado pelo método GET, POST.)

Exemplo:

```
session_start();  
$_SESSION["NOME"] = "Rogerio";
```

Em todas as páginas que é utilizado a sessão, a mesma tem que ser iniciada ou seja, usar o comando **SESSION_START()**



Quando é iniciado uma sessão é gerado um **ID** para cada **usuário**. Ele não se repete




```
<?php
    session_start();
    echo "Id da sessão: ".SESSION_ID();
?>
```

A função **SESSION_ID** retorna o **ID** do **usuário** na sessão




A função **session_regenerate_id()** permite criar um novo id para o usuário



```
session_start();  
session_regenerate_id();  
echo "Id da sessão: ".SESSION_ID();
```

A função **session_save_path()** mostra o path onde é salvo os arquivos de sessão.



```
session_start();  
echo "path: ".session_save_path();
```

STATUS DA SESSÃO

SESSION_STATUS

✓ Retorna o status atual da sessão

```
$id_status = session_status();
```


STATUS DA SESSÃO

O Valor retornado são três tipos:

PHP_SESSION_DISABLED




- ✓ Se as sessões estiverem desabilitadas. Equivale a zero

PHP_SESSION_NONE

- ✓ Se as sessões estiverem habilitadas, mas nenhum existir. Equivale a 1

PHP_SESSION_ACTIVE

- ✓ Se as sessões estiverem habilitadas e uma existir. Equivale a 2.

```
$id_status = session_status();  
switch($id_status) {  
    case 0:  Representa o PHP_SESSION_DISABLED  
        echo "As sessões estão desabilitadas ";  
        break;  Representa o PHP_SESSION_NONE  
    case 1:  
        echo "As sessões estão habilitadas, mas não existem ";  
        break;  
    case 2:  Representa o PHP_SESSION_ACTIVE  
        echo "As sessões estão habilitadas e uma existe ";  
        break;  
}
```

✓ **unset()**

Exclui uma variável de sessão.

Exemplo:

```
unset($_SESSION['txtnome']);
```

```
unset($_SESSION['txtsenha']);
```

No exemplo acima está excluindo as variáveis de sessão TXTNOME e TXTSENHA

✓ **\$_SESSION**

São chamadas variáveis de sessão onde são armazenados dados, principalmente informações do usuário.

Exemplo:

```
$_SESSION['txtnome'] = $login;
```

```
$_SESSION['txtsenha'] = $senha;
```

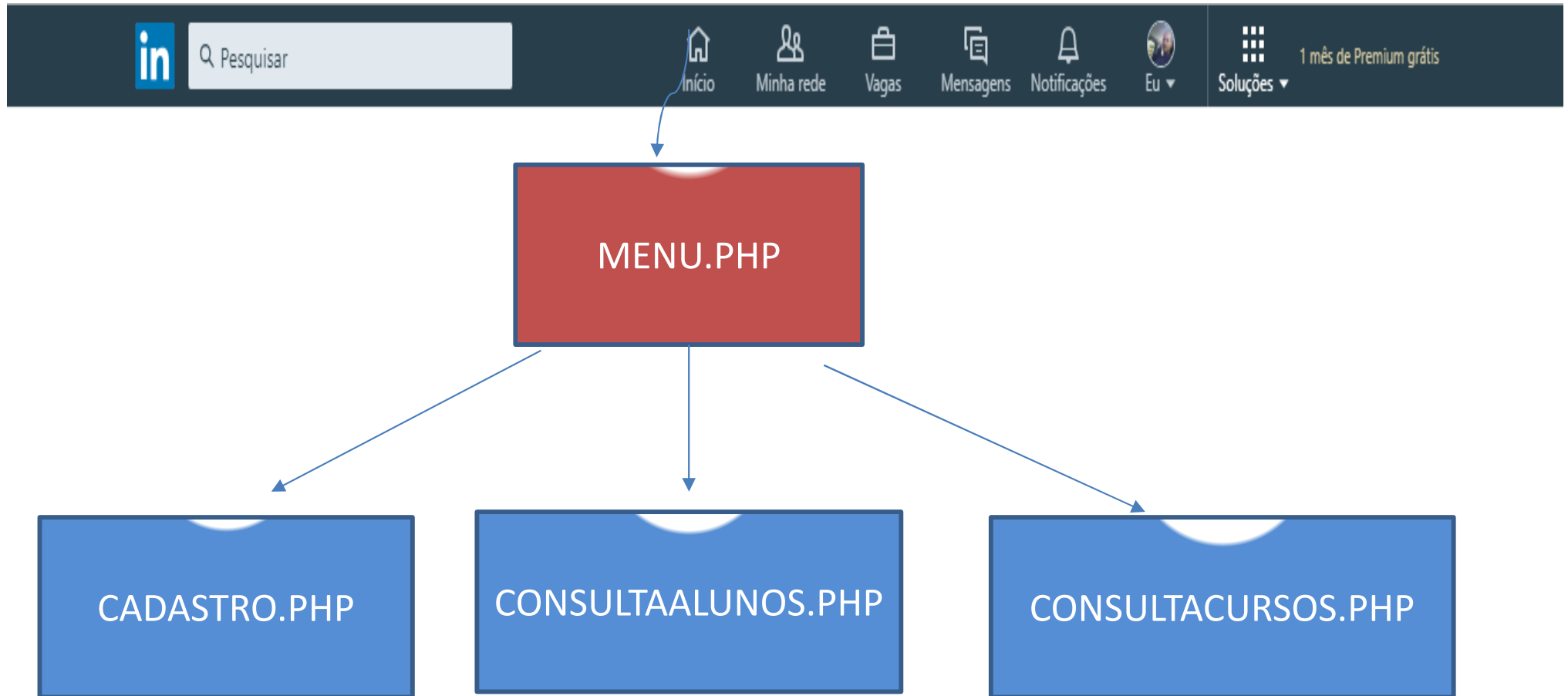
✓ ISSET()

Esta função ISSET serve para verificar se a função existe. Retorna true se a variável existe e retorna falso se caso não existe.

Exemplo com função ISSET()

```
<?php
    session_start();
    if((!isset($_SESSION['txtnome']) == true) and (!isset($_SESSION['txtsenha']) == true))
    {
        unset($_SESSION['txtnome']);
        unset($_SESSION['txtsenha']);
        header('location:index.html');
    }
    $logado = $_SESSION['txtnome'];
?>
```

Incorporando scripts com include, include_once, require, require_once



INCLUDE, INCLUDE_ONCE

INCLUDE('nome arquivo') – É um construtor do PHP que permite incluir uma script de código quantas vezes for se implementado no programa.

INCLUDE_ONCE('nome arquivo') – É um construtor do PHP que permite incluir uma script de código somente uma vez.

O Construtor **INCLUDE** ou **INCLUDE_ONCE** irá emitir um **WARNING(AVISO)** caso não encontre o arquivo especificado para gerar a script mas não interrompe a execução do programa.

REQUIRE, REQUIRE_ONCE

REQUIRE('nome arquivo') – É um construtor do PHP que permite incluir uma script de código quantas vezes for se implementado no programa.

REQUIRE_ONCE('nome arquivo') – É um construtor do PHP que permite incluir uma script de código somente uma vez.

O Construtor **REQUIRE** ou **REQUIRE_ONCE** irá emitir um **FATAL ERROR(ERRO FATAL** caso não encontre o arquivo especificado para gerar a script e interrompe a execução do programa.

Funções para manipulação de arquivos

A linguagem PHP nos oferecem dezenas de funções para trabalhar com sistema de arquivos. Principais operações que pode-se realizar sobre um arquivo são: Abertura, Leitura, Escrita e Fechamento.

PARÂMETRO	DESCRIÇÃO
NOME ARQUIVO	Nome do arquivo a ser aberto, que pode ser local ou remoto
MODO	Modo de acesso ao arquivo.
USAR_INCLUDE_PATH	Indica se o arquivo deve ser procurado nas pastas especificadas na diretiva <u>include_path</u> do <u>php.ini</u> .
CONTEXTO	Permite a definição de um contexto, que consiste em um conjunto de parâmetros que modificam o comportamento do arquivo.

O parâmetro ***nome_arquivo*** pode referenciar um arquivo que está no mesmo computador ou em um computador remoto. Se caso o parâmetro iniciar com *http://*, será aberta uma conexão HTTP, que retornará um ponteiro para o arquivo que será aberto. Se o nome do arquivo iniciar com *ftp*, será aberta uma conexão FTP antes da abertura do arquivo.

O Segundo parâmetro da função *fopen* é o modo, que pode possuir os seguintes valores:

Parâmetros da função OPEN

MODO	DESCRIÇÃO
'r'	Abre somente para leitura, posicionando o ponteiro no início do arquivo.
'r+'	Abre para leitura e escrita, posicionando o ponteiro no início do arquivo.
'w'	Abre somente para escrita, posicionando o ponteiro no início do arquivo e deixando com tamanho zero.
'w+'	Abre para leitura e escrita, posicionando o ponteiro no início do arquivo deixando-o com tamanho zero.
'a'	Abre somente para escrita, posicionando o ponteiro no final do arquivo. Se o arquivo não existir, tenta criá-lo.
'a+'	Abre para leitura e escrita, posicionando o ponteiro no final do arquivo. Se o arquivo não existir, tenta criá-lo.
'x'	Cria e abre um arquivo somente para escrita, posicionando o ponteiro no início do arquivo. Se o arquivo já existir, retorna falso(FALSE) e gera um erro do tipo E_WARNING.
'x+'	Cria e abre um arquivo para leitura escrita, posicionando o ponteiro no início do arquivo. Se o arquivo já existir, retorna falso(FALSE) e gera um erro do tipo E_WARNING.

- **FCLOSE**

Para fechar um arquivo, é utilizado a função FCLOSE, que possui a seguinte sintaxe:

Bool fclose(recurso ponteiro_arquivo).

Retorna true ou false se o arquivo foi fechado com sucesso e retorna false se houver alguma falha. O parâmetro passado para a função *fclose* deve conter a variável para a qual foi atribuído o resultado da função *FOPEN*

- Exemplo:

<?

```
$ponteiro = fopen('arquivo.txt', 'r')
```

```
Echo $ponteiro;
```

```
fclose($ponteiro);
```

?>

FREAD

Uma das formas de ler dados de um arquivo é utilizar a função *FREAD*, que permite especificar a quantidade de informações a serem lidas.

Sintaxe:

String fread(recurso ponteiro arquivo, int tamanho)

Exemplo:

<?

```
$ponteiro = fopen("teste.txt", "r");
```

```
$conteudo = fread($ponteiro,30);
```

```
Echo $conteudo;
```

```
Fclose($ponteiro);
```

?>

FGETS

Também realiza a leitura de um arquivo, mas lê no máximo uma linha. É bastante utilizado quando queremos trabalhar individualmente com cada linha do arquivo. Sua sintaxe é a seguinte:

```
String fgets(recurso ponteiro_arquivo [, int tamanho]).
```

FGETS

A leitura é feita até que seja lido o número de bytes especificados em tamanho, ou quando terminar a linha atual do arquivo(caracter\n), ou quando o arquivo chegar ao seu final. Se não for especificado o parâmetro tamanho, será utilizado o valor padrão que é 1.024 bytes.

Exemplo:

<?

```
$ponteiro = fopen("teste.txt", "r");
```

```
$linha = fgets($ponteiro, 4096);
```

```
echo $linha;
```

```
fclose($ponteiro);
```

?>

FWRITE

Para escrever dados em um arquivo com o PHP, utilizamos a função `fwrite`, que possui a seguinte sintaxe:

```
Int fwrite(recurso ponteiro_arquivo, string  
string[int tamanho])
```

Exemplo:

<?

```
$conteudo = "Este texto será escrito no  
arquivo";
```

```
$ponteiro = fopen("arquivo.txt", "w");
```

```
Fwrite($ponteiro, $conteudo);
```

```
Fclose($ponteiro);
```

?>

Conexão com banco de dados

Exemplo

<?

```
$conexao=mysql_connect("localhost", "root", " ") or  
die ("Conexão não efetuada");
```


Onde:

localhost – é o endereço do servidor, onde o banco está armazenado;

- **root** – é o usuário do banco de dados;
- string [**senha**] – é a senha do usuário do banco de dados MySQL. No exemplo anterior está como vazio.
- **die** – parâmetro opcional que exibe uma mensagem indicando que a conexão não foi efetuada.

- `mysql_select_db("nomebanco")`
- Exemplo:
- `mysql_select_db('senaisul');`
- No exemplo acima **SENAISUL** é o banco de dados.

- Conforme os exemplos apresentados para conexão com o banco então poderia ser feito dessa forma:

```
<?php
    mysql_connect ('localhost', 'root',' ') or
    die(mysql_error());
    mysql_select_db("senaisul");
?>
```

EXEMPLO DE INCLUSÃO DE DADOS

```
<?php
mysql_connect ('localhost', 'root', '') or die(mysql_error('Erro na conexao'));
mysql_select_db('senaisul');
mysql_query("SET NAMES 'utf8'");
// Passando os dados obtidos pelo formulário para as variáveis abaixo
$curso      = $_POST['txtcurso'];
$cargahor   = $_POST['txtcargahoraria'];
$nr vagas   = $_POST['txtvagas'];
$result     = "INSERT INTO curso (ds_curso,vl_cargahoraria,nr_vagas)
              VALUES ('$curso','$cargahor','$nr vagas')";
mysql_query($result);
mysql_close();
?>
```

Leitura de registros

```
<?php
mysql_connect ('localhost', 'root', '') or die(mysql_error('Erro na conexao'));
mysql_select_db('senaisul');
mysql_query("SET NAMES 'utf8'");
$resultado = mysql_query("select * from cadastro");
?>
```

O comando SELECT faz uma consulta na tabela cadastro

Procedimento para mostrar registros

```
<?php

while($reg_cadastro=mysql_fetch_array($resultado))
{
    $codigo=$reg_cadastro["id_codigo"]."<br>";
    $nome=$reg_cadastro["ds_nome"]."<br>";
    $dsendereco=$reg_cadastro["ds_endereco"]."<br>";
    $nmCidade=$reg_cadastro["ds_cidade"]."<br>";
}

?>

<tr>
    <td><?php echo $codigo ?></td>
    <td><?php echo $nome ?></td>
    <td><?php echo $dsendereco ?></td>
    <td><?php echo $nmCidade ?></td>
    <td align="center"><a href="editar.php?id=<?php echo $reg_cadastro["id_codigo"]?>"></img></a>
    <td align="center"><a href="#" onclick = "apagar('<?php echo $reg_cadastro["id_codigo"]
        ?>');"></a></td>

</tr>

<?php
```

Procedimento para filtrar por determinado registro.

```
<?php
mysql_connect ('localhost', 'root', '') or die(mysql_error('Erro na conexao'));
mysql_select_db('senaisul');

if(isset($_GET["id"])){
    if(is_numeric($_GET["id"])){
        $sql = "select * from cadastro where id_codigo = ".$_GET["id"];
        $executa = mysql_query($sql);
        $resultado = mysql_fetch_array($executa);
    }
}
?>
```

Procedimento para excluir registro.

```
<?php
mysql_connect ('localhost', 'root','') or die(mysql_error('Erro na conexao'));
mysql_select_db('senaisul');
if(is_numeric($_GET["id"])){
    $sql= "delete from cadastro where id_codigo = ".$_GET["id"];
    $query = mysql_query($sql);
    if(mysql_affected_rows() > 0 ){
        echo "<script>alert('Registro apagado com sucesso');</script>";
        echo "<script>window.location='consulta.php'</script>";
    }
}
?>
```


Procedimento para alterar registro.

```
<?php
mysql_connect ('localhost', 'root','') or die(mysql_error('Erro na conexao'));
mysql_select_db('senaisul');
mysql_query("SET NAMES 'utf8'");

$codigo    = $_POST["id_codigo"];
$nome      = $_POST["txtnome"];
$endereco  = $_POST['txtendereco'];
$cidade    = $_POST["txtcidade"];

$SQL = "update cadastro set ds_nome='$nome',ds_endereco='$endereco',ds_cidade='$cidade' where
        id_codigo = '$codigo'";

$executa = mysql_query($SQL);
echo "<script>alert('registro alterado');</script>";
echo "<script>window.location='consulta.php'</script>";
?>
```

Referências

Linguagem PHP. Disponível em: <http://www.php.net>. Acesso em: 02/02/2017.

Melo, Alexandre Altair Nascimento Maurício. PHP Profissional: Aprenda a desenvolver sistemas profissionais orientados a objeto com padrões de projeto. São Paulo. Editora: Novatec. 2008.

Soares Wallace. PHP5: Conceitos, programação e integração com Banco de Dados. Editora: Érica. 2004.