Aulas PHP do Básico ao Intermediário

Este documento oferece suporte à série de videoaulas do curso de PHP básico ao intermediário. Aqui estão alguns exemplos funcionais além de imagens ilustrando o conteúdo apresentado em aula.

Autor: Gunnar Correa
E-mail: gunnercorrea@gmail.com
Site: www.gunnarcorrea.com

Sumário

Softwares

Netbeans: https://netbeans.org/downloads

Visual Studio Code: https://code.visualstudio.com

Notepad++: https://notepad-plus-plus.org/download/v7.3.3.html

Sublime Text: https://www.sublimetext.com

XAMPP: https://www.apachefriends.org/download.html

WAMP: http://www.wampserver.com/en

Referências

Wikipédia: https://pt.wikipedia.org/wiki/PHP

PHP Manual: https://secure.php.net/manual/pt_BR/index.php

W3C: http://www.w3schools.com/php/

MSDN: https://msdn.microsoft.com/en-us/library/gg276466.aspx

Objetivos

O curso tem como objetivo principal apresentar as funcionalidades básicas e conteúdos necessários para que qualquer iniciante consiga seguir em frente com a linguagem, utilizando recursos mais sofisticados como envio de e-mail, Banco de Dados e outros recursos.

Conteúdos

1. O que é o PHP
2. Imprimindo texto
3. Variáveis e Comentários
4. Operadores matemáticos
5. If, Else e Elseif
6. Switch
7. While
8. For
9. Dowhile
10. Array
11. Foreach
12. Function
13. Funções do PHP
14. Post
15. Get
16. Data e hora
17. Ler e gravar em TXT
18. Requisitando arquivos
19. Criptografias
20. Cookie
21. Session

O que é o PHP

PHP (um acrônimo recursivo para "PHP: Hypertext Preprocessor", originalmente Personal Home Page) é uma linguagem interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo dinâmico no World Wide Web. Figura entre as primeiras linguagens passíveis de inserção em documentos HTML, dispensando em muitos casos o uso de arquivos externos para eventuais processamentos de

dados. O código é interpretado no lado do servidor pelo módulo PHP, que também gera a página web a ser visualizada no lado do cliente. A linguagem evoluiu, passou a oferecer funcionalidades em linha de comando, e além disso, ganhou características adicionais, que possibilitaram usos adicionais do PHP, não relacionados a web sites. É possível instalar o PHP na maioria dos sistemas operacionais, gratuitamente. Concorrente direto da tecnologia ASP pertencente à Microsoft, o PHP é utilizado em aplicações como o MediaWiki, Facebook, Drupal, Joomla, WordPress, Magento e o Oscommerce.

Criado por Rasmus Lerdorf em 1995, o PHP tem a produção de sua implementação principal — referência formal da linguagem, mantida por uma organização chamada The PHP Group. O PHP é software livre, licenciado sob a PHP License, uma licença incompatível com a GNU General Public License (GPL) devido a restrições no uso do termo PHP.

Figura 1: Rasmus Lerdorf (Fonte: https://pt.wikipedia.org/wiki/PHP)

Imprimindo Texto

Echo – Imprime um valor solicitado.

Print – Imprime um valor solicitado e retorna 1, é possível atribuir o valor em uma variável.

```
1.  echo “Meu nome é Cida”;
2.  echo “Meu nome é: {$nome}”;
```

```
3.  echo “Meu nome é: ” . $nome;
4.
5.  print “Meu nome é Cida”;
```

Variáveis e Comentários

Variáveis servem para armazenar valores que serão utilizados posteriormente, seu valor fica armazenado na memória RAM.

PHP é uma linguagem **Case Sensitive**, onde diferencia caracteres maiúsculos de minúsculos.

Bin	Oct	Dec	Hex	Sinal
0100 0000	100	64	40	@
0100 0001	101	65	41	A
0100 0010	102	66	42	B
0100 0011	103	67	43	C
0100 0100	104	68	44	D
0100 0101	105	69	45	E
0100 0110	106	70	46	F

Bin	Oct	Dec	Hex	Sinal
0110 0000	140	96	60	`
0110 0001	141	97	61	a
0110 0010	142	98	62	b
0110 0011	143	99	63	c
0110 0100	144	100	64	d
0110 0101	145	101	65	e
0110 0110	146	102	66	f

PHP possui Tipagem automática ou conhecida como fracamente ‘Tipado’, porém é importante conhecer os tipos de variáveis e os valores que poderão ser atribuídos, veja abaixo uma tabela com as principais variáveis.

Tipo	**Utilização**	**Exemplo**
String	Utilizada para armazenar textos	$nome = “SatellaSoft”;
Char	Utilizado para armazenar um único caractere	$sexo = ‘M’;
Float	Utilizado para armazenar números quebrados	$salario = 2500.00
Int	Utilizado para armazenar números inteiros	$idade = 23;
bool	Armazena apenas Verdadeiro ou Falso	$ativo = true; $ativo = false;

Para se criar uma variável basta utilizar o símbolo de dólar, em seguida o nome, é importante ressaltar que não se devem usar acentos nem caracteres especiais com exceção do underline (_).

```
1.  $nome_do_cliente = “Maria”;
```

É possível deixar comentários no código fonte, estes comentários não serão interpretados.

```
1.  // - apenas uma linha.
2.  /**/ - comenta um bloco.
```

Operadores matemáticos

Os seguintes operadores poderão ser utilizados sem o uso de funções adicionais.

Operador	Nome
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão

Fonte: http://php.net/manual/pt_BR/language.operators.arithmetic.php

Exemplo

```
1.  $a = 5;
2.  $b = 5;
3.  $c = $a + $b;
4.  echo $c;
5.  //Resultado é 10.
```

If, Else e Elseif

Operador condicional IF permite perguntar se algo é válido, podendo perguntar se uma variável existe e até mesmo perguntar qual o valor de uma variável.

Sua sintaxe é:

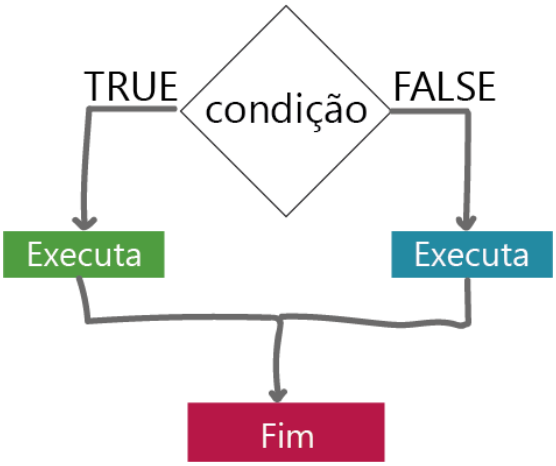
```
1.  If($nome == “SatellaSoft”){
2.     echo “Válido”;
3.  }else{
4.     echo “Inválido”;
```

```
5.  }
```

O If permite uma série de comparações que é exibido na tabela abaixo.

Valor	Nome	Utilização
==	Igual	If($a == $b){}
===	Idêntico nome e tipo	If($a === $b){}
!=	Diferente	If($a != $b){}
!==	Não idêntico tipo ou valor	If($a !== $b){}
<>	Diferente	If($a <> $b){}
<	Menor	If($a < $b){}
>	Maior	If($a > $b){}
>=	Maior ou igual	If($a >= $b){}
<=	Menor ou igual	If($a =< $b){}

A condição IF retorna **true** caso uma condição seja satisfeita e **false** caso contrário.



Podemos utilizar mais de uma condição no if, que são elas:

Valor	Nome	Utilização
&&	And (e)	If($a < $b && $a > 5){}
||	Or (ou)	If($a < $b || $b != 10){}

Além do IF e ELSE existe o ELSEIF que adiciona uma condição ao else, podendo assim fazer outra comparação antes de executar algo.

```
1.  If(X){
2.  }elseif(x == y){
3.
4.  }
```

Switch

Basicamente muito parecido com o IF, porém é destinada a fazer comparações de igualdade, sua sintaxe é:

```
switch($valor){
  case “1”:
    echo “bom dia”;
   break;
case “2”:
    echo “boa tarde”;
   break;
default
    echo “boa noite”;
}
```

Fonte: http://php.net/manual/pt_BR/control-structures.switch.php

While

Com estrutura de repetição é possível utilizar as condições de comparação do IF. Necessita que a satisfação seja concluída para sair do laço.

Utiliza-se o While apenas quando não sabemos quando será satisfeito o critério de saída do While.

Exemplo:

```
$cont = 1;
While($count <=10){
   echo $count;
   count++;
}
```

For

A sintaxe do for é diferente do While, porém sua funcionalidade é a mesma. Utiliza-se o for quando sabemos a quantidade de vezes que o laço vai se repetir, como exibir os anos de 1992 até 2016.

O for assume três parâmetros

```
for(par1; par2; par3){
   echo par3;
}
```

Primeiro parâmetro: Executa apenas uma única vez, utilizado para atribuir um valor.
Segundo parâmetro: Condição de saída, para cada iteração esta

condição é validada.

Terceiro parâmetro: No final de cada iteração este parâmetro é executado.

Veja um exemplo:

```
1.  for($i = 0; $i < 10; $i++){
2.     echo “Número” . $i
3.  }
```

Definição de iteração:
https://pt.wikipedia.org/wiki/Itera%C3%A7%C3%A3o

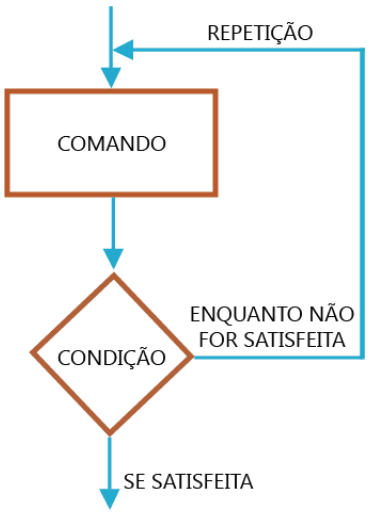
DoWhile

Este com certeza é um dos laços mais interessantes do PHP, basicamente ele “**atira primeiro e pergunta depois**”. Primeiro ele executa uma instrução para posteriormente verificar a condição de saída.

A condição fica no **While**, porém a repetição dentro do **Do,** veja o exemplo a seguir.

```
1.  $i = 1;
2.  do{
3.      echo "Número {$i}";
4.      $i++;
5.  }while($i <= 10);
```

Fonte: http://www.w3schools.com/php/php_looping.asp



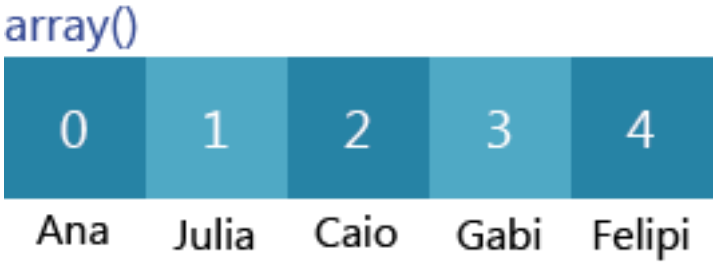
Array

Array são estruturas capazes de criar lista de valores, algo como pilhas, dicionário e vetor. Imagine uma variável quebrada em vários pedaços, armazenando vários valores.

Em linguagens como **C#** e **Java**, é possível criar um vetor para armazenar apenas um único tipo de dado, como **INT**, **STRING** ou **CHAR**.

Um array possui um índice, que nada mais é do que a posição de leitura, sempre começando do 0, ou seja, se for armazenar cinco nomes, então teremos a sequência de índices, 0, 1, 2, 3 e 4.

Veja a figura a seguir.



Para ler certo valor, podemos basicamente acessar através de seu índice.

Veja o Array a seguir.

```
1.  $nomes = array("Ana", "Julia", "Caio", "Gabi", "Felipi");
2.  echo $nomes[3];
3.  //Resultado: Gabi
```

Podemos utilizar o for para exibir estes valores, veja outro exemplo a seguir.

```
1.  for($i = 0; $i < 5; $i++){
2.     echo "Nome: {$nomes[$i]} <br/>";
3.  }
```

Podemos utilizar um Array Associativo, em Javascript é chamado de Objeto Literal e em C# de Objeto Anônimo.

```
1.  $nomes = array( 1 => "Ana", 2 => "Gabi");
```

Este Array associativo pode ser convertido facilmente para um JSON.

Podemos também inserir um Array dentro de outro Array, veja o seguinte exemplo.

```php
$notas = array(
        "ana" => array(
            "1bim" => 7,
            "2bim" => 10,
        ),
        "gabi" => array(
            "1bim" => 10,
            "2bim" => 10,
        ),
        "julia" => array(
            "1bim" => 4,
            "2bim" => 5,
        ),
    );
    echo $notas["julia"]["1bim"] . "<br/>";
    echo $notas["julia"]["2bim"];
```

Existem outras formas de se utilizar o Array, como usar o array_push() que insere um valor no Array conforme a sua necessidade. Consulte a documentação do PHP para obter mais informações.

Fonte: http://php.net/manual/pt_BR/language.types.array.php

Foreach

Agora que você já entendeu como utilizar laços de repetição e Array/vetores, podemos ver o último laço de repetição, que tem uma tarefa muito interessante e diferente dos demais.

O foreach é responsável por fazer uma consulta em uma lista e retornar todo conteúdo de uma respectiva linha. Imagine uma tabela com varias linhas e colunas, o Foreach traz o valor de cada linha até não haver mais linhas.

Assista a animação de 23 segundos que apresenta como o foreach funciona, https://www.youtube.com/watch?v=BslbK5dvQ8k.

Veja como é utilizado.

```php
$nomes = array("Ana", "Julia", "Caio", "Gabi", "Felipi");
foreach($nomes as $valor){
    echo "Nome atual: {$valor} <br/>";
}
```

O primeiro parâmetro é a lista de valores, podendo ter várias linhas e colunas, em seguida o **as**, que faz atribuição ao parâmetro seguinte.

Function

Quando se desenvolve um projeto muito grande, é possível ter várias partes que necessitam do mesmo código, então ficar duplicando códigos pode dar muito trabalho na hora de fazer a manutenção e deixar o projeto mais pesado.

Para solucionar vários problemas existem as funções, elas criam blocos de instruções que podem retornar valores, receber valor e com isto executar ações.

Usamos **function(parametros){ ações}**, veja um exemplo abaixo.

```
1.  function exibirNome(){
2.      echo "Meu nome é Paulo";
3.  }
4.  echo exibirNome();
```

No exemplo proposto, criamos uma função que exibe o nome, no código a seguir vamos passar o nome que queremos exibir.

```
1.  function exibirNome($nome){
2.      echo "Meu nome é {$nome}";
3.  }
4.  echo exibirNome("Ana");
```

As funções possuem um recurso chamado de **return**, que retorna um valor, ou seja, depois de tudo executado podemos definir que a função deve me retornar o valor de uma soma por exemplo.

Podemos passar vários valores por parâmetro, mas a ordem deve ser respeitada, tanto quando for chamar quanto atribuir na função.

Veja o exemplo.

```
1.  function soma($a, $b){
2.      return $a + $b;
3.  }
4.  echo soma(10, 15);
5.  //soma (10 = $a, 15 = $b);
```

Funções do PHP

strlen()

Descrição: Esta função retorna à quantidade de caracteres de uma string, na qual devemos passar por parâmetro. Link: http://php.net/manual/pt_BR/function.strlen.php

```
1.  <?php
```

```
2.      $meuNome = "Gunnar Correa";
3.      $quantidadeCaracteres = strlen($meuNome);
4.      echo "Meu nome contém {$quantidadeCaracteres} caracteres.";
5.  ?>
```

substr()

Descrição: Esta função retorna uma parte de uma string, na qual podemos informar o início e término ou apenas de onde se deve iniciar. Link: http://php.net/manual/pt_BR/function.substr.php

```
1.  <?php
2.      $meuNome = "Gunnar Correa";
3.      $primeiroNome = substr($meuNome, 0, 6);
4.      echo "Meu primeiro nome é <b>{$primeiroNome}</b>.";
5.  ?>
```

strtolower()

Descrição: Esta função converte todos os caracteres da string para minúscula. Link: http://php.net/manual/pt_BR/function.strtolower.php

```
1.  <?php
2.      $meuNome = strtolower("Gunnar Correa");
3.      echo "Meu nome é <b>{$meuNome}</b>.";
4.  ?>
```

strtoupper()

Descrição: Esta função converte todos os caracteres da string para maiúscula. Link: http://php.net/manual/pt_BR/function.strtoupper.php

```
1.  <?php
2.      $meuNome = strtoupper("Gunnar correa");
3.      echo "Meu nome é <b>{$meuNome}</b>.";
4.  ?>
```

strip_tags()

Descrição: Esta função remove tags HTML de uma string. Link: http://php.net/manual/pt_BR/function.strip-tags.php

```
1.  <?php
```

```
2.      $comentario = "<b>Gunnar Correa</b> -
     Você pode conferir um exemplo em   <a       href='http://www.google.com'>G
    oogle</a>";
3.      echo $comentario;
4.      echo "<br />";
5.    echo strip_tags($comentario, "<b>");
6.  ?>
```

str_replace/str_ireplace()

Descrição: Esta função remove determinada ocorrência de toda uma string. O str_replace diferencia maiúscula de minúscula e o str_ireplace não faz essa diferenciação. Link: http://php.net/manual/pt_BR/function.str-replace.php, http://php.net/manual/pt_BR/function.str-ireplace.php

```
1.  <?php
2.      $mensagen = "Acesse o site do <b>PHP</b>";
3.      $procurar = "PhP";
4.      $substituir = "Google";
5.      echo str_replace($procurar, $substituir, $mensagen);
6.      echo "<br/>";
7.      echo str_ireplace($procurar, $substituir, $mensagen);
8.  ?>
```

explode()

Descrição: Esta função quebra uma string em um ponto especificado e retorna um array, na qual podemos acessar um de seus índices. Link: http://php.net/manual/pt_BR/function.explode.php

```
1.  <?php
2.      $meuNome = "Gunnar Correa Pereira";
3.      $nomeArray = explode(" ", $meuNome); //Quebra nos 'espaços'
4.      echo "Primeiro nome: <b>{$nomeArray[0]}</b>,
5.      último nome: <b>{$nomeArray[2]}</b>";
6.  ?>
```

POST

Post é um array associativo utilizado para enviar informações para o cabeçalho da página. Quando submetemos uma página, o valor deste array é passado para o cabeçalho da página e assim poder acessar o valor contido.

O Post é muito utilizado em para obter os valores do formulário e passarmos para o servidor, como por exemplo, um formulário de cadastro ou consulta.

Podemos trabalhar de várias formas para obter os valores do formulário e passar para o servidor, aqui serão apresentadas duas formas, que são:

$_POST["nomeCampo"] – Muito comum e bastante utilizada, porém não oferece uma segurança a nossa aplicação, pois a outra sintaxe nos permite filtrar os valores de entrada.

filter_input(INPUT_POST, "nomeCampo", FILTER_SANITIZE_STRING) - Pouco utilizada, porém nos fornece mais segurança e controle das informações que estão sendo passadas ao nosso servidor. Podemos passar vários valores para ela, como o terceiro, por exemplo, informamos qual será o conteúdo de entrada, neste caso apenas textos, ignorando caracteres especiais.

Tipos de filtros: https://www.w3schools.com/php/php_ref_filter.asp

GET

Ele faz o contrário do POST, ao invés de passarmos algum valor, o get obtém informações. Ele também é um array associativo e é processado no cabeçalho da nossa página.

Podemos trabalhar de várias formas para obter os valores com o GET, aqui serão apresentadas duas formas, que são:

$_GET[""] – Muito comum e bastante utilizada, porém não oferece uma segurança a nossa aplicação, pois a outra sintaxe nos permite filtrar os valores de entrada.

filter_input(INPUT_GET, "nomeCampo", FILTER_SANITIZE_STRING) - Pouco utilizada, porém nos fornece mais segurança e controle das informações que estão sendo requisitadas. Podemos passar vários valores para ela, como o terceiro, por exemplo, informamos qual será o conteúdo de entrada, neste caso apenas textos, ignorando caracteres especiais.

Tipos de filtros: https://www.w3schools.com/php/php_ref_filter.asp

Data e hora

Está função é utilizada para formatarmos a data e hora de acordo com os valores informados, e assim ela nos retorna uma string.

Sua utilização é bem simples, como mostra o exemplo abaixo.

```
1.  <?php
2.        echo date(“d/m/Y h:i:s”); Saída: 28/04/2017 13:45:14
3.  ?>
```

Podemos formatar os valores de saída para que ele exiba de outras formas, para isto consulte a documentação em: http://php.net/manual/pt_BR/function.date.php

Por padrão o PHP vai obter a data e hora do servidor, porém podemos definir qual é o fuso horário que o servidor deve se basear, para isto antes da utilização do date, usamos o **date_default_timezone_set('America/Sao_Paulo')**.

Lista de fuso horário suportado: http://php.net/manual/pt_BR/timezones.php

Ler e gravar em TXT

Para fazer a persistência das informações em um arquivo, como TXT, por exemplo, vamos utilizar algumas funções que estão listadas abaixo. É muito importante saber que se um arquivo estiver sendo utilizado por uma aplicação, ele não poderá ser alterado por outra até que ele seja fechado.

Ciclo de vida para persistência de um arquivo

Abrir
Ler/Gravar
Fechar

fopen() – abre um arquivo para ser manipulado:
http://php.net/manual/pt_BR/function.fopen.php

fread() – Com base no ponteiro definido no fopen(), está função faz a leitura. http://php.net/manual/pt_BR/function.fread.php

fwrite() – com base no ponteiro definido no fopen(), está função escreve no arquivo. http://php.net/manual/pt_BR/function.fwrite.php

fclose() – Fecha o ponteiro de um arquivo.
http://php.net/manual/pt_BR/function.fclose.php

Gravando

```php
<?php
//Criamos uma função que recebe um texto como parâmetro.
function gravar($texto){
    //Variável arquivo armazena o nome e extensão do arquivo.
    $arquivo = "meu_arquivo.txt";
    //Variável $fp armazena a conexão com o arquivo e o tipo de ação.
    $fp = fopen($arquivo, "a+");
    //Escreve no arquivo aberto.
    fwrite($fp, $texto);
    //Fecha o arquivo.
    fclose($fp);
}
gravar("Olá, mundo");
?>
```

Lendo

```php
<?php
//Criamos uma função que irá retornar o conteúdo do arquivo.
function ler(){
    //Variável arquivo armazena o nome e extensão do arquivo.
    $arquivo = "meu_arquivo.txt";
    //Variável $fp armazena a conexão com o arquivo e o tipo de ação.
    $fp = fopen($arquivo, "r");
    //Lê o conteúdo do arquivo aberto.
    $conteudo = fread($fp, filesize($arquivo));
    //Fecha o arquivo.
    fclose($fp);
    //retorna o conteúdo.
    return $conteudo;
}
echo ler();
?>
```

Requisitando arquivos

Durante todo desenvolvimento de nossa aplicação precisamos trabalhar com vários arquivos, cada um deles possuem diferentes tipos de responsabilidades e também de funcionalidades, para carregar estes arquivos ao nosso projeto é muito simples, basta declarar o tipo de requisição e passar a URL na qual queremos incluir ao nosso projeto, veja abaixo os tipos de requisições:

include – requisita um arquivo, caso não seja localizado, exibe uma mensagem de *warning* e continua a execução do script;

include_once – requisita um arquivo, caso não seja localizado, exibe uma mensagem de *warning* e continua a execução do script, porém se o arquivo já foi requisitado, ele não carrega novamente.

require – requisita um arquivo, caso não seja localizado, exibe uma mensagem de *error* e interrompe a execução do script.

Require_onde – requisita um arquivo, caso não seja localizado, exibe uma mensagem de *error* e interrompe a execução do script, porém se o arquivo já foi requisitado, ele não carrega novamente.

Podemos utilizar de várias formas, como mostra o script abaixo.

```php
<?php
//include
include("data.php");
include('data.php');
include"data.php";
include'data.php';

//include_once
include_once("data.php");
include_once('data.php');
include_once"data.php";
include_once'data.php';

//require
require("data.php");
require('data.php');
require"data.php";
require'data.php';

//require_once
require_once("data.php");
require_once('data.php');
require_once"data.php";
require_once'data.php';

?>
```

Para este exemplo, vamos utilizar dois arquivos, o arquivo **data.php** e o arquivo **index.php**, veja os exemplos abaixo.

data.php

```php
<?php
echo date("d/m/Y h:i:s");
?>
```

index.php

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        require_once("data.php");
        ?>
    </body>
</html>
```

Criptografias

Para garantir a segurança das informações persistidas no banco de dados ou em uma transição de URLs, podemos implementar vários tipos de criptografias.

Sha 1: Gera uma chave sem volta com 40 caracteres.
http://php.net/manual/pt_BR/function.sha1.php
Md5: Gera uma chave sem volta com 32 caracteres.
http://php.net/manual/pt_BR/function.md5.php
Base64_encode: Criptografa uma string, porém permite retornar ao valor original. http://php.net/manual/pt_BR/function.base64-encode.php
Base64_decode: Retorna o valor original de uma string criptografada em base64. http://php.net/manual/pt_BR/function.base64-decode.php

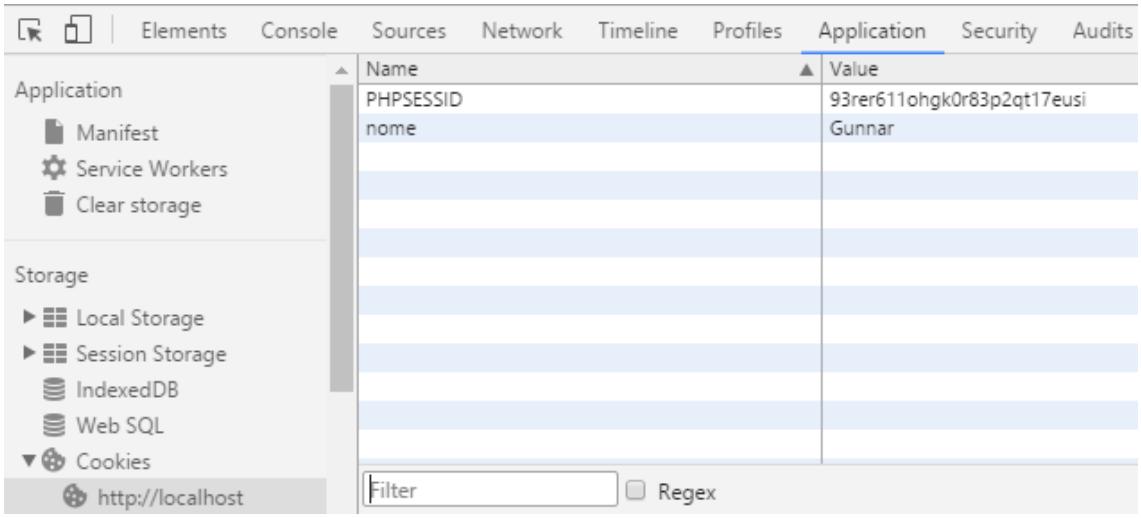
```
//SHA 1
$s = sha1("Olá, mundo");
echo $s;

echo "<br/>";

//MD5
$m = md5("PHP intermediário");
echo $m;

echo "<br/>";

//Base64_encode
$b = base64_encode("criptografias");
echo $b;
echo "<br/>";
echo base64_decode($b);
```

Cookie

Cookie é um mecanismo que permite guardar informações no lado do cliente (navegador), estás informações são geradas no cabeçalho da página, então tem que estar antes de qualquer outra saída.

Documentação sobre cookies: http://php.net/manual/pt_BR/features.cookies.php

```
1.  //Criamos uma cookie
2.  setcookie("nome","valor", tempo_de_vida);
3.
4.  //Obter o valor de uma cookie
5.  $_COOKIE["nome"];
6.
7.  //Alterar o valor de uma cookie
8.  setcookie("nome","valor_novo");
9.
10. //Deletar o cookie
11. setcookie("nome","valor", time() - 1);
```



Session

Sessão é muito semelhante aos cookies, porém ao invés da informação ficar alocada no cliente (navegador), ela fica armazenada no servidor. Toda sessão cria uma Cookie, pois o servidor não sabe para quem ele está guardando aquela informação, então ele cria um cookie e sempre que o cliente for requisitar algo, ele informa o código da sessão que está gravado na cookie.

Como sessão é um recurso que consome espaço e processamento do servidor, por padrão ela vem desabilitada.

Session_start: Inicia uma sessão para o host atual.
http://php.net/manual/pt_BR/function.session-start.php

$_SESSION[“nome”] = ‘’: Cria uma sessão e atribui um valor a ela.
http://php.net/manual/pt_BR/reserved.variables.session.php

Session_destroy: Destrói todas as sessões para o host atual.
http://php.net/manual/pt_BR/function.session-destroy.php

```php
session_start();

$_SESSION["nome"] = "Ana";

echo $_SESSION["nome"];

session_destroy();
```