COM222 DESENVOLVIMENTO DE SISTEMAS WEB

Aula 09: PHP – Parte I

Conteúdo da aula

- Introdução à linguagem PHP

  histórico, estrutura de um arquivo PHP, execução de PHP, Informações do ambiente.

- Sintaxe básica

  variáveis, operadores, estruturas condicionais e de repetição.

- Funções PHP
- Escopo de variáveis e Super Globais
- Inclusão de Arquivos
- Como trabalhar com

  Formulários HTML, cookies, arquivos, horas e datas.

- Como criar um validador básico para entrada de dados

PHP

- **Desenvolvida em 1995 por Rasmus Lerdorf (membro do Grupo Apache)**
  - Originalmente projetada para fazer tracking dos visitantes do site de Lerdorf
  - Em 2 anos, passou a ser largamente utilizada em conjunto com o servidor Apache
  - Evoluiu para uma linguagem de scritpt poderosa para programação server-side
  - Gratuita e de código aberto
  - Possui plug-ins para vários servidores
  - Possui facilidades para trabalhar com bases de dados mySQL

- **PHP é uma linguagem até certo ponto parecida com JavaScript, porém server-side**
  - Código PHP é embutido em HTML através de tags
  - Quando uma requisição de página chega, o servidor reconhece conteúdo PHP através da extensão de arquivo (`.php` ou `.phtml`)
  - O servidor executa o código PHP, gerando saídas na página HTML de resposta
  - A página resultante é então enviada ao cliente
  - O usuário nunca vê código PHP, apenas a saída HTML na página de resposta

Informação sobre o PHP

A função `phpinfo()` provê informação sobre o ambiente PHP -- versão instalada no servidor, pararâmetros usados na instalação, etc.

```
<html><head></head>
<!- info.php -->
<body>
<?php
// Mostra toda a informção PHP
phpinfo();
?>
<?php
// Mostra apenas informações gerais
phpinfo(INFO_GENERAL);
?>
</body>
</html>
```

INFO_GENERAL	The configuration line, php.ini location, build date, Web Server, System and more
INFO_CREDITS	PHP 4 credits
INFO_CONFIGURATION	Local and master values for php directives
INFO_MODULES	Loaded modules
INFO_ENVIRONMENT	Environment variable information
INFO_VARIABLES	All predefined variables from EGPCS
INFO_LICENSE	PHP license information
INFO_ALL	Shows all of the above (default)

Sintaxe básica do PHP

Um bloco de script PHP sempre começa com `<?php` e termina com `?>`. Um bloco de código PHP pode ser colocado (quase) em qualquer lugar de um documento HTML.

```
<html>
<!-- hello.php -->
<head><title>Hello World</title></head>
<body>
 <p>Isto será ignorado pelo interpretador PHP.</p>

   <?php echo '<p>Já isto será interpretado.</p>'; ?>

  <p>Isto também será ignorado.</p>

   <?php print('<p>Olá e bem vindo à <i>minha</i> página!</p>');
   $var = 'isso é uma variável!';
   ?>
   <?='Podemos imprimir valores assim também!'?>
   <?=$var?>
  <?php
   //Isto é um comentário
  /*
   Isto é um
   bloco de
   comentário
  */
  ?>
</body>
</html>>>
```

`print`, `echo` e a tag especial `<?= ?>`

usados para saída

`Ponto e vírgula (;)` no fim de cada comando PHP

O servidor executa os comando print e echo, substituindo o resultado na saída

Variáveis escalares

Todas as variáveis em PHP começam com o símbolo $. O tipo de uma variável é determinado de acordo com o contexto no qual ela é usada (tipagem fraca).

```
<html><head></head>
<!-- escalares.php -->
<body>  <p>
<?php
$foo = true; if ($foo) echo "Valor TRUE! <br /> \n";
$txt='1234'; echo "$txt <br /> \n";
$a = 1234; echo "$a <br /> \n";
$a = -123;
echo "$a <br /> \n";
$a = 1.234;
echo "$a <br /> \n";
$a = 1.2e3;
echo "$a <br /> \n";
$a = 7E-10;
echo "$a <br /> \n";
echo 'Arnold disse: "I\'ll be back"', "<br /> \n";
$beer = 'Bohemia';
echo "Uma $beer agora cairia bem<br /> \n";
$str = <<<EOD
Exemplo de uma string
em múltiplas linhas
usando a sintaxe “heredoc”.
EOD;
echo $str;
?>
</p>
</body>
</html>
```

Quatro tipos escalares:

boolean
  true or false

integer,

float,
 ponto flutuante

string
 aspas simples
 aspas duplas

Arrays

Um array é de fato um mapa ordenado – tipo que relaciona valores para chaves.

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

`array()` = cria um array

key = um inteiro ou uma string.

value = qualquer tipo PHP.

```
<?php
array(5 => 43, 32, 56, "b" => 12);
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Se nenhuma chave é fornecida (como no exemplo), o interpretador PHP usa (maior dos índices + 1).

Se for uma chave existente, seu valor é sobrescrito.

```
<?php
$arr = array(5 => 1, 12 => 2);
foreach ($arr as $key => $value) { echo $key, '=>',
                                        $value;  }
$arr[] = 56;    // o mesmo que $arr[13] = 56;
$arr["x"] = 42; // adiciona um novo elemento
unset($arr[5]); // remove o elemento
unset($arr);    // deleta todo o array
$a = array(1 => 'one', 2 => 'two', 3 => 'three');
unset($a[2]);
$b = array_values($a);
?>
```

Para modificar valores

`unset()` remove um par chave valor

`array_values()` faz reindexação do array (indexação numérica)

Constantes

Uma constante é um identificador (nome) para um valor. Constantes são case-sensitive por padrão. Por convenção, constantes devem aparecer em letras maiúsculas.

```php
<?php

// Constantes válidas
define("FOO", "algo");
define("FOO2",    "algo diferente");
define("FOO_BAR", "algo ainda mais diferente");

// Constantes inválidas  (não podem começar
//      com um número!)

define("2FOO",    "something");

// Isso é válido, mas deve ser evitado:
// PHP pode um dia criar uma constante que
// inutilize seu script

define("__FOO__", "something");

?>
```

É possível acessar uma constante em qualquer lugar no código (não há questões de escopo)

Constantes

- Podemos testar se as constantes estão definidas usando a função defined("NOME_DA_CONSTANTE"). Retorna 1 (true) para declarada e 0 (false).

```
<?php
define('NAME','value');
defined('name') OR exit('Não foi definida');
echo 'isso não será executado';
?>
```

- Pode-se encerrar o script caso a constante não esteja definida usando exit() ou die() (que semanticamente significam a mesma coisa). Esse tipo de pratica é comum em frameworks para evitar acesso direto a arquivos importantes.

Operadores

- **Operadores aritméticos:** +, -, *,/ , %, ++, --
- **Operadores de atribuição:** =, +=, -=, *=, /=, %=

Isto	é o mesmo que
x+=y	x=x+y
x-=y	x=x-y
x*=y	x=x*y
x/=y	x=x/y
x%=y	x=x%y

- **Operadores de comparação:** ==, !=, >, <, >=, <=,===,!==

  === e !=== comparam se o valor e o tipo são o mesmo

- **Operadores lógicos:** &&, ||, !

- **Operadores de string**: . e .= (para concatenação de strings)

```
$a = "Hello ";
$b = $a . "World!"; // agora $b contém"Hello World!"
$a = "Hello ";
$a .= "World!";
```

Estruturas condicionais: if else

Execução de bloco de código depende que uma condição seja satisfeita

```
<html><head></head>
<!-- if-cond.php -->
<body>

<?php
$d=date("D");
echo $d, "<br/>";
if ($d=="Fri")
     echo "Bom fim de semana! <br/>";
else
     echo "Bom dia! <br/>";
?>

</body>
</html>
```

```
<html><head></head>
<!-- if-cond.php -->
<body>

<?php
$d=date("D");
echo $d, "<br/>";
if ($d=="Fri"):?>
  <p>Bom fim de semana!<p><br/>
<?php else:?>
  <p>Bom dia!</p><br/>
<?php endif;?>

</body>
</html>
```

Existem duas formas para realizar a condição. Apesar de apresentarem o mesmo efeito, a segunda opção em geral é mais elegante quando estamos lidando com saídas HTML.

Estruturas condicionais: if else

```
if (condição){
// Algo acontece
}
else{
//Algo acontece
}
```

```
if(condição):
// Algo acontece
else:
// Algo acontece
endif;
```

date() é uma função interna do PHP que pode ser chamada com diferentes parâmetros para retornar a data (e/ou hora) local em vários formatos

No caso do nosso exemplo, é retornada uma string de 3 letras representando o dia da semana.

Deem uma olhada no manual da linguagem para verificar os formatos possíveis:
http://php.net/manual/en/function.date.php

O MANUAL DA FERRAMENTA PODE AJUDAR MUITO NO APRENDIZADO!!!!
https://secure.php.net/manual/pt_BR/index.php

Estruturas condicionais: switch

Permite escolher uma entre diversas opções de blocos de código

```
<html><head></head>
<body>
<!-- switch-cond.php -->
<?php
$x = rand(1,5);  // número aleatório
echo "x = $x <br/><br/>";
switch ($x)
{
case 1:
  echo "Número 1";
  break;
case 2:
  echo "Número 2";
  break;
case 3:
  echo "Número 3";
  break;
default:
  echo "Nenhum nro entre 1 e 3";
  break;
}
?>

</body>
</html>
```

```
switch (expressão)
{
case label1:
   código a se executado se
   expressão = label1;
   break;
case label2:
   código a se executado se
   expressão = label2;
   break;
default:
   código a se executado se
   expressão é diferente de label1
   e label2
   break;
}
```

Laços: while e do-while

Bloco de código do laço é executado dependendo de uma condição.

```
<html><head></head>
<body>

<?php
$i=1;
while($i <= 5)
{
  echo "O número é $i <br />";
  $i++;
}
?>

</body>
</html>
```

Executa o laço se, e enquanto, a condição especificada for verdadeira

```
<html><head></head>
<body>

<?php
$i=0;
do
{
  $i++;
  echo "O número é $i <br />";
}
while($i <= 10);
?>

</body>
</html>
```

Executa o bloco de código do laço uma vez e, continua a repetir enquanto a condição for verdadeira

Laços: for e foreach

Loop depende de um contador

```php
<?php
for ($i=1; $i<=5; $i++)
{
   echo "Hello World!<br />";
}
?>
```

Repete a execução de um bloco de código o número de vezes especificado no laço

```php
<?php
$a_array = array(1, 2, 3, 4);
foreach ($a_array as $value)
{
   $value = $value * 2;
   echo "$value <br/> \n";
}
?>
```

```php
<?php
$a_array=array("a","b","c");
foreach ($a_array as $key => $value)
{
  echo $key . " = " . $value . "\n";
}
?>
```

Repete a execução de um bloco de código para cada elemento de um array

Funções de usuário

Pode-se definir funções utilizando a seguinte sintaxe:

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
   echo "Exemplo de função.\n";
   return $retval;
}
?>
```

É possível definir funções condicionais, funções que chamam outras funções e funções recursivas.

Função pode retornar um valor de qualquer tipo

```
<?php
function square($num)
{
   return $num * $num;
}
echo square(4);
?>
```

```
<?php
function small_numbers()
{
   return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
echo $zero, $one, $two;
?>
```

```
<?php
function takes_array($input)
{
   echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
 takes_array(array(1,2));
?>
```

Escopo de variáveis

O escopo de uma variável é o contexto no qual ela foi definida.

```
<?php
$a = 1; /* escopo limitado */
function Test()
{
    echo $a;
  /* referência a uma var de escopo local
*/
}
Test();
?>
```

O escopo é local dentro de funções, assim, o valor de $a é indefinido no comando “echo”.

```
<?php
$a = 1;
$b = 2;
function Sum()
{
    global $a, $b;
    $b = $a + $b;
}
Sum();
echo $b;
?>
```

`global`

Refere-se a versão global da variável.

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
Test();
Test();
Test();
?>
```

`static`

não perde valor.

Super Globais

Existem variáveis chamadas SUPER GLOBAIS que estão disponíveis em qualquer escopo. (ou seja qualquer arquivo pode acessar).

$GLOBALS	**array com todas as globais no escopo atual**
$_SERVER	**cabeçalhos, paths e localização de scripts**
$_GET	**captura os parâmetros de uma requisição GET**
$_POST	**captura os parâmetros de uma requisição POST**
$_FILES	**array de itens enviados pelo método POST**
$_COOKIE	**array com COOKIES recebidos do cliente**
$_SESSION	**array com variáveis de sessão**
$_REQUEST	**array com $_COOKIE, $_GET e $_POST**

Super Globais

Vamos explorar os valores do array $_SERVER e ver o que o servidor pode nos mostrar.

```
<html><head></head>
<body>

<?php
if (isset($_SERVER['HTTP_REFERER'])) {
    echo "Referer: " . $_SERVER["HTTP_REFERER"] . "<br />";
}
echo "Browser: " . $_SERVER["HTTP_USER_AGENT"] . "<br />";
echo "IP address: do usuário" . $_SERVER["REMOTE_ADDR"];
?>

<?php
echo "<br/><br/><br/>";
echo "<h2>Toda informação</h2>";
foreach ($_SERVER as $key => $value)
  {
    echo $key . " = " . $value . "<br/>";
  }
?>

</body>
</html>
```

Tente explorar as outras SUPER GLOBAIS e veja as informações que podemos ter.

https://secure.php.net/manual/pt_BR/language.variables.superglobals.php

Inclusão de arquivos

```php
//  index.php
<?php

$var = 'valor';
global $g = 'valor global';
include('test.php');
require_once('test.php');
// repare que o require_once não executa
// os comandos echo
?>

// test.php
<?php

echo $GLOBALS['g']; // valor global
echo $var; // valor

?>
```

Podemos incluir arquivos de várias formas

include('file_path'.'file_name.php'); ou

require('file_path'.'file_name.php');

Inclui o arquivo que herda as variáveis do que o chamou, executa, e retorna para terminar o anterior.

include_once('file_path'.'file_name.php'); ou

require_once('file_path'.'file_name.php');

Inclui o arquivo **SE ELE JÁ NÃO FOI INCLUIDO**.

*O escopo de variáveis em arquivos incluídos depende da posição do comando "include", ou seja ele possui as variaveis do arquivo que o chamou.

Formulários

Elementos de formulário ficam automaticamente disponíveis utilizando variáveis PHP (desde que o elemento HTML tenha um "name" definido).

```
<html><head></head>
    <!-- form.html -->
    <body>
        <form action="welcome.php" method="post">
            Digite seu nome: <input type="text" name= "nome" /> <br/>
            Digite sua idade: <input type="text" name= "idade" /> <br/>
            <input type="submit" value="POST" /> <input type="reset" />
        </form>
        <a href="welcome.php">GET</a>
    </body>
</html>
```

```
<html><head></head>
    <!-- welcome.php -->
    <body>
        <?php if ($_SERVER['REQUEST_METHOD'] == 'POST'): ?>
            Você usou o metodo POST!<br/>
            Bem vindo <?php echo $_POST["nome"] . "."; ?><br/>
            Você tem <?php echo $_POST["idade"]; ?> anos!
        <?php else: ?>
            Voce usou o metodo GET!<br/>
        <?php endif; ?>
    </body>
</html>
```

`$_POST`

contém dados de um POST.

`$_GET`

contém dados de um GET.

Cookies

`setcookie(name,value,expire,path,domain)` cria um cookie.

```
<?php
if (isset($_COOKIE['cookie_name']))
    setcookie('cookie_name', $_COOKIE['cookie_name'] + 1, time() + 36000);
else
    setcookie('cookie_name', 1, time() + 36000);
?>
<html>
    <body>
        <p>
            Prezado Cliente, um cookie foi setado nesta página! O cookie ficará ativo
quando o cliente enviá-lo de volta ao servidor. Seu valor sera alterado a cada visita na
página
        </p>
        <p> Valor do Cookie:<b>(<?= $_COOKIE['cookie_name'] ?>)</b> </p>
    </body>
</html>
```

setcookie()

deve aparecer ANTES de <html> uma vez que deve fazer parte do header da página.

`$_COOKIE`

contém os dados do cookie.

`isset()`

Checa se foi declarada a variável.

nome do cookie pode ser usado como variável

Validador básico de Entradas

Script multipropósito que solicita informação de contato e verifica se campos obrigatórios foram informados.

pages/confirma.php

```
<html><head><meta charset="utf-8"></head><body>
<h2>Obrigado! Seguem as informações digitadas.</h2>
<h3>Informações de contato</h3>
<p>Name:<?=$_POST['f_name']?> <?=$_POST['l_name']?></p>
<p>Email:<?=$_POST['email']?></p>
<p>OS:<?=$_POST['os']?></p>
</body></html><!-- Esta página sera aberta se tudo ocorrer bem -->
```

pages/form.php

```
<html><head><meta charset="utf-8"></head><body>
<form action="index.php" method="post">
Nome: <input type="text" name="f_name" value="<?php echo get_post_value($_POST['f_name'])?>"
/> <br/>
Sobrenome<b>*</b>:<input type="text" name="l_name" value="<?php echo
get_post_value($_POST['l_name'])?>" /> <br/>
Email<b>*</b>:<input type="text" name="email" value="<?php echo
get_post_value($_POST['email'])?>" /> <br/>
Sistema Operacional: <input type="text" name="os" value="<?php echo
get_post_value($_POST['os'])?>" /> <br/><br/>
<input type="submit" name="submit" value="Submit" /> <input type="reset" />
</form></body></html>
```

ATENÇÂO: CRIE O DIRETÓRIO pages NO DIRETÓRIO RAIZ DE SUA APLICAÇÂO!!

Check and Confirm Functions

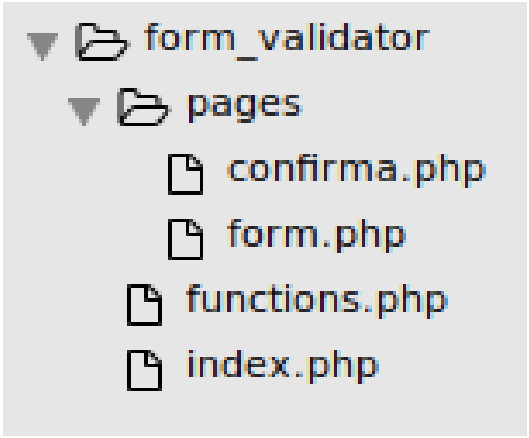
Crie o arquivo functions.php no diretório raiz de sua aplicação

```
<?php

// arquivo functions.php com funções de verificação!

function get_post_value($var) {
if(isset($var)) return $var;
else return '';
}
function validate_form() {
if($_POST['l_name'] and $_POST['email'])
        return true;
echo '<h3>Campos obrigatórios não foram preenchidos!</h3>';
return false;
}
```

Programa principal

Esse é o arquivo index.php que ficará na pasta raiz da aplicação

```
<?php
define('PAGE_FOLDER','pages/');
include_once('functions.php');

if($_SERVER['REQUEST_METHOD'] == 'POST') {
    if(validate_form()){
        require_once(PAGE_FOLDER.'confirma.php');
    } else {
        require_once(PAGE_FOLDER.'form.php');
    }
} else {
    include_once(PAGE_FOLDER.'form.php');
}
```



A estrutura de seus arquivos deve estar parecida com essa, se estiver ocorrendo algum erro verifique os nomes e se estão nas pastas corretas.

Exercício 1 – Entrega: hoje

Altere o exercício anterior.

1)Valide o email – só aceite strings que contenham “@”

2)Inclua o campo ano de nascimento. Só aceite valores entre 1950 e 2000.

3)Não permita o acesso direto a arquivos que não sejam o index.php e apresente uma mensagem caso haja a tentativa

4)Informe ao usuário que digitar os campos corretamente os nomes, ips e horário de acesso de outros usuários.

Ajuda para a parte 1: http://www.w3schools.com/php/php_string.asp

Manipulando Bases de Dados MySQL via PHP

Funções PHP e mySQL podem ser usadas em conjunto

- Conectar ao servidor e realizar login (este é o comando PHP)

```
$my = new mysqli("host","username","password",”database”)
```

Host:	localhost
Database:	escola
Username:	root
Password:	----

- Envie queries SQL para adicionar, modificar ou remover dados

```
$res = my->query("query")
```
(mesma string usada no console SQL, porém sem o ponto e vírgula final)

- Pegue os resultados da query (SELECT)

```
while ($row = $res->fetch_assoc()) echo $row[“COLUNA”];
```

- Feche a conexão com o servidor de banco de dados

```
my->close();
```

Exercício – Inserir e consultar registros no banco

- Criar base de dados escola
  - CREATE DATABASE escola;

- Criar tabela aluno;
  - CREATE TABLE aluno(matric INT NOT NULL,
    nome VARCHAR(48),
    endereco VARCHAR(48),
    email VARCHAR(48),
    PRIMARY KEY(matric));

Inserir registros: data_in.php

```
<html><head><title>Inserindo dados no banco</title></head>
<body>
<?php
/*insert students into DB*/
if($_SERVER['REQUEST_METHOD'] == 'POST')  {
   $db = new mysqli('localhost','root','','escola');
   $sql="INSERT INTO aluno  VALUES(" . $_POST["matric"] . ",'" .
   $_POST["nome"] . "','" . $_POST["endereco"] . "','" . $_POST["email"] .
   "')";
   echo $sql;
   $db->query($sql); /*  executa a query  */
   $db->close();
   echo"<h3>Obrigado. Seus dados foram inseridos</h3> \n";
   echo'<p><a href="data_in.php">Inserir outro aluno</a></p>' . "\n";
   echo'<p><a href="data_out.php">Veja a lista de alunos</a></p>' ."\n";
   }  else  {
?>
<h3>Entre com seus dados</h3>
<form action="index.php" method="post">
    Matricula: <input type="text" name="matric" /> <br/>
    Nome......: <input type="text" name="nome" /> <br/>
    Endereco.: <input type="text" name="endereco" /> <br/>
    email.......: <input type="text" name="email" /> <br/>
    <br/>
    <input type="submit" name="submit" /> <input type="reset" />
</form>
<?php }?>
</body></html>
```

Consultar registros: data_out.php

```
<html><head><title>Consulta alunos</title></head><body>
<h1> Banco de dados de alunos</h1>
<p> Ordenar a lista de estudantes por
<a href="data_out.php?ordem=matric">matricula</a>,
<a href="data_out.php?ordem=nome">nome</a>, or
by <a href="data_out.php?ordem=email">e-mail</a>.
</p><?php/* obtem alunos do banco */
$db = new mysqli("localhost", "root",'','escola');
if (!isset($_GET["ordem"]))
            $sql = "SELECT * FROM aluno";
        else {
            switch ($_GET["ordem"]) {
                case 'matric': $sql = "SELECT * FROM aluno ORDER BY matric";
                    break;
                case 'nome': $sql = "SELECT * FROM aluno ORDER BY nome";
                    break;
                case 'email': $sql = "SELECT * FROM aluno ORDER BY email";
                    break;

                default: $sql = "SELECT * FROM aluno";
                    break;
            }
        }
        $result = $db->query($sql);       /*  execute a query */
while ($row = $result->fetch($result)) {
    echo "<h4> Nome: " . $row["nome"] . "</h4> \n";
    echo "<h5> Matricula: " . $row["matric"] . "<br/> Email: " . $row["email"] .
   "<br/> Endereco: " . $row["endereco"] . "</h5> \n";
}
$db->close();
?></body></html>
```

Exercício 2 – Entrega: hoje

1)Elabore uma página PHP que permita consultar os dados de um estudante, dado seu número de matrícula.

2)Elabore uma página PHP que permita atualizar os dados de um estudante, dado seu número de matrícula.