Aula 3º

Estruturas Avançadas II

Olá,

Seguiremos por mais uma aula da matéria Desenvolvimento Voltado à Web I. Na aula anterior, foi feito um estudo sobre os arrays e arquivos, exemplificando como manipular inserindo e retornando informações armazenadas nessas estruturas.

Na de hoje, veremos mais algumas estruturas, tais como GET e POST, para receber informações do lado do usuário, as funções e escopo que as delimitam, além dos Cookies e Sessions, que armazenam informações do usuário no lado do cliente e do servidor, respectivamente.

Leia atentamente a esta aula e, se tiver alguma dúvida, use os recursos que estão na sua área do aluno.

Bons estudos!

Objetivos de aprendizagem

Ao término desta aula, vocês serão capazes de:

- enviar dados do lado do cliente para o servidor via GET e POST;
- criar funções para melhorar o código e entender conceitos de escopo;
- armazenar informações no lado do cliente para posterior recuperação (Cookies);
- armazenar informações no lado do servidor para posterior recuperação (Sessions).

Seções de estudo

1 – Obtendo informações do Usuário (GET e POST)
2 – Funções e Escopo
3 – Cookies e Sessions

1- Obtendo informações do Usuário (GET e POST)

Em Silva (2014), o PHP é apresentado como uma ótima ferramenta para a manipulação de formulários, que em sistemas web é uma ótima maneira para interagir com o usuário. Isso garante uma maior interatividade. Já que ele fará todas as entradas necessárias, o servidor recupera essas informações e retorna o resultado para o lado do cliente.

Nesta seção serão descritos a criação de um formulário, os métodos de envio de informação e suas características, e a manipulação e retorno das informações.

Subseção A – Formulário HTML

Para exemplificar como receber informações do usuário replicaremos o formulário encontrado em IMC (2019). O site criou uma calculadora para calcular o peso ideal de cada pessoa, através do IMC, que é o índice de massa corpórea.

Iremos criar um formulário para que o usuário informe seu nome, sua altura e seu peso. Essas informações serão processadas no lado do servidor que retorna para o usuário seu IMC, sua classificação e seu grau de obesidade. Os respectivos valores podem ser encontrados na Figura 1.

VEJA A INTERPRETAÇÃO DO IMC		
IMC	CLASSIFICAÇÃO	OBESIDADE (GRAU)
MENOR QUE 18,5	MAGREZA	0
ENTRE 18,5 E 24,9	NORMAL	0
ENTRE 25,0 E 29,9	SOBREPESO	I
ENTRE 30,0 E 39,9	OBESIDADE	II
MAIOR QUE 40,0	OBESIDADE GRAVE	III

Figura 1 – Interpretação do IMC. Fonte: IMC, 2019.

Então, o primeiro passo a se fazer é a construção do formulário.

```
<!DOCTYPE html>
<html lang="pt-br">
   <head>
         <title>IMC</title>
   </head>
   <body>
         <form>
               Nome:          <input type="text" name="nome"><br>
               Altura:        <input type="number" name="altura"><br>
               Peso:          <input type="number" name="peso"><br>
               <input type="submit">
         </form>
   </body>
</html>
```

O formulário possui 3 *inputs*, um do tipo texto para o nome e dois do tipo número, para a altura e o peso. Além disso, possui um *input* do tipo submit que é o botão para executar a ação do formulário.

Subseção B – Métodos de envio (GET e POST)

Quando criamos um formulário no HTML podemos alterar alguns atributos do seu elemento, o method e action. Enquanto o primeiro indica a forma de envio para o servidor, o segundo indica qual a página que será chamada no servidor para receber as informações.

```
<!DOCTYPE html>
<html lang="pt-br">
   <head>
         <title>IMC</title>
   </head>
   <body>
         <form          method="POST" action="imc.php">

         </form>
   </body>
</html>
```

Dessa forma, então, indicamos que o formulário será enviado por POST e a página que receberá a informação é a imc.php. Podemos fazer com que a resposta do servidor seja dada pela mesma página que contém o formulário. Para isso, podemos utilizar uma variável superglobal do PHP, para preencher o atributo action do formulário.

```
<form method="POST" action="<?php echo $_SERVER['PHP_SELF'];?>">
</form>
```

A variável superglobal $_SERVER é um array que contém várias informações úteis para o servidor. Através da chave 'PHP_SELF' conseguimos o caminho para o próprio arquivo que está sendo executado.

O atributo method do formulário aceita dois valores, GET ou POST.

GET

Quando utilizamos esse método os parâmetros são passados no cabeçalho da requisição e são visíveis na URI (ALURA, 2019).

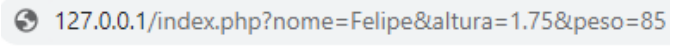

Figura 2 – Método GET. Fonte: Acervo pessoal.

O problema de utilizar esse método é a limitação de tamanho (em geral permite no máximo 255 caracteres) e também os problemas com segurança. Este segundo ocorre porque as informações são enviadas junto da URI. Em nosso exemplo as informações não são tão sensíveis, mas se fosse preciso enviar login e senha não seria muito indicado utilizar

o método GET (SOARES, 2014).

Esse método é mais recomendado quando se faz necessária a obtenção de um determinado recurso, como formulários de busca e listagem de produtos que precisam ser mais performáticos (ALURA, 2019).

POST

Já o método POST não envia as informações aparecendo na URI, mas não quer dizer que é a maneira mais segura de se enviar. Para isso, é necessário utilizar o HTTPS que possibilita a criptografia dos dados enviados (ALURA, 2019).

Esse método permite o envio de informações maiores como imagens e arquivos. No nosso exemplo, ambos os métodos resolvem o nosso problema, mas vamos trabalhar com o POST.

Subseção B – Tratando as informações

Formulário construído, método escolhido, vamos então tratar as informações que serão recebidas e gerar o resultado final para exibir ao usuário.

No início do nosso arquivo, faremos com que o PHP verifique se alguma informação já foi enviada por método POST (utilizaremos a chave 'REQUEST_METHOD' do array $_SERVER). Caso ainda não tenha sido enviado, o nosso formulário é exibido para o usuário; caso o usuário já tenha enviado, com o botão exibiremos, antes do formulário, as informações de valor do IMC, classificação e grau de obesidade.

Armazenaremos em um array $arr as informações do nome, altura e peso, que serão obtidas pela função *filter_input*, que recebe essa variável externa pelo seu nome e a filtra para que não seja enviado algo que possa quebrar nosso programa.

O cálculo do IMC é a divisão do peso do paciente pelo quadrado de sua altura. O resultado do IMC é armazenado dentro do array $imc["valor"], e, então, nesse mesmo array são armazenadas as informações de classificação $imc["classificacao"] e grau de obesidade $imc["grau"].

Antes de exibir o formulário é realizada uma verificação com o array $imc["valor"]. Com a função isset() ela verifica se a variável passada por parâmetro já foi setada no programa, isso vai ocorrer quando $imc["valor"] já estiver preenchido com o resultado. Caso já esteja preenchido, são exibidas as informações relativas ao IMC. Caso contrário, só o formulário será exibido.

```
<?php
    if($_SERVER['REQUEST_METHOD']==='POST'){
        $arr["nome"]= filter_input(INPUT_POST, 'nome', FILTER_SANITIZE_SPECIAL_CHARS);
    $arr["altura"]= filter_input(INPUT_POST, 'altura', FILTER_SANITIZE_SPECIAL_CHARS);
    $arr["peso"]= filter_input(INPUT_POST, 'peso', FILTER_SANITIZE_SPECIAL_CHARS);
        $imc["valor"] = $arr["peso"]/$arr["altura"]*$arr["altura"];
            if($imc["valor"]<18.5){
$imc["classificacao"]="MAGREZA";
                $imc["grau"]="0";
    }elseif($imc["valor"]<24.9){
    $imc["classificacao"]="NORMAL";
                $imc["grau"]="0";
    }elseif($imc["valor"]<29.9){
    $imc["classificacao"]="SOBREPESO";
                $imc["grau"]="I";
    }elseif($imc["valor"]<39.9){
    $imc["classificacao"]="OBESIDADE";
                $imc["grau"]="II";
    }else{
    $imc["classificacao"]="OBESIDADE GRAVE";
                $imc["grau"]="III";
    }
    }
    ?>
    <!DOCTYPE html>
    <html lang="pt-br">
        <head>
            <title>IMC</title>
        </head>
        <body>
            <?php
                if(isset($imc["valor"])){
                    echo "<h3>O usuário: ".$arr["nome"]." com peso: ".$arr["peso"]." E altura: ".$arr["altura"]."</h3>";
    echo "<h3>Foi classificado como: ".$imc["classificacao"]." Grau de obesidade: ".$imc["grau"]."</h3>";

                }
            ?>
            <form>
                Nome: <input type="text" name="nome"><br>
                Altura: <input type="number" name="altura"><br>
                Peso: <input type="number" name="peso"><br>
                <input type="submit">
            </form>
        </body>
    </html>
```

2- Funções e Escopo

A reutilização de código está presente cada vez mais nos softwares modernos. Como exemplo, podemos criar uma função de geração de relatório genérico que pode ser utilizado tanto por um sistema web de fluxo de caixa, quanto para um sistema web de controle de alunos de uma escola. Cria-se, então, uma função que possa ser reutilizada diversas vezes.

Mas a utilização de funções leva à necessidade de entender um novo conceito, o escopo, pois quando dentro da função, a execução do código assume um novo escopo, que pode gerar alguns problemas quando não implementado corretamente.

Subseção A – Função

Soares (2014) define uma função como subprograma, que contém uma série de instruções que serão executadas, podendo retornar ou não um valor. Esse programa pode ser chamado diversas vezes e onde quer que seja no código.

Para exemplificar a criação e utilização das funções, vamos retomar os conceitos do IMC. Será descrita a criação de uma função que, dependendo do parâmetro tipo, retorna o valor do IMC, ou a classificação, ou o grau de obesidade.

A criação da função é feita através da palavra reservada function, seguida do nome da função (esse nome deve ser único em todo o script). Entre parênteses devem ser informados os argumentos da função e seu código deve ser inserido entre chaves. Seguindo a sintaxe abaixo:

```
<?php
   function imc($altura,$peso){
        r e t u r n $peso/$altura*$altura;
}
?>
```

No exemplo acima, o único retorno da função é o cálculo do IMC. Podemos, então, criar o parâmetro tipo, que por padrão terá o valor 0 (retorna o valor do IMC), o valor 1 retorna a classificação e o valor 2 retorna o grau de obesidade.

```
<?php
   f u n c t i o n imc($altura,$peso,$tipo=0){
        $imc=$peso/$altura*$altura;
if($imc<18.5){
            $class = "MAGREZA";
$grau = "0";
}elseif($imc 24.9){
$class = "NORMAL";
$grau = "0";
}elseif($imc <29.9){
$class = "SOBREPESO";
$grau = "I";
}elseif($imc <39.9){
$class = "OBESIDADE";
$grau = "II";
}else{
$class = "OBESIDADE GRAVE";
$grau = "III";
}

if($tipo===0){
            return $imc;
        }elseif($tipo===1){
            return $classificacao;
}elseif($tipo===2){
   return $grau;
}
}
?>
```

O valor padrão permite que o argumento que tenha um padrão seja opcional na hora da chamada da função. O argumento, então, assume o valor padrão para a execução da função.

O resultado da função pode assumir qualquer valor, seja inteiro, ponto flutuante, booleano, string, array. Desde que esteja corretamente definido após a palavra reservada return (SOARES, 2014).

Um ponto interessante a se destacar é que não importa onde a função esteja declarada dentro do script do PHP. Ela pode ser criada, inclusive, depois de uma chamada da função.

Subseção B – Escopo

O escopo nada mais é que o contexto em que são definidas as variáveis. Ele delimita que variável pode ser manipulada em cada trecho de código.

Exceto nas funções e classes, existe apenas um escopo do script. As variáveis declaradas nele podem ser vistas por todo o script, inclusive nos arquivos incluídos e requeridos.

Quando a variável é declarada dentro de uma função, ela está delimitada a ser utilizada apenas dentro dessa função, limitando-se ao escopo local da função.

Para que uma variável declarada no escopo do script possa ser acessada dentro de uma função, devemos utilizar a palavra reservada global para instanciar as variáveis dentro da função. Dessa forma, a variável local pega a referência da versão global de todas as variáveis declaradas.

```
<?php
   $valor = 6;
   function dobro(){
        $valor = 10;
        return valor * 2;
}
echo $valor." e ".dobro();
//6 e 20
?>
```

Nesse primeiro exemplo, a variável $valor fora da função é diferente da variável $valor dentro da função. Portanto, mesmo que eu altere o valor da variável dentro da função, ela nada se relaciona com a variável fora da função.

```
<?php
   $valor = 6;
   function dobro(){
        global $valor;
$valor = 10;
        return valor * 2;
}
echo $valor." e ".dobro();
//10 e 20
?>
```

Nesse segundo exemplo, com a utilização da palavra reservada global, fazemos uma referência à variável $valor que está fora da função. Dessa forma, o valor alterado dentro da função será alterado também fora da função.

3- Cookies e Sessions

Os cookies e sessions são variáveis que possibilitam o compartilhamento de informação entre diversas páginas de um mesmo site. A partir do momento em que o cookie ou a sessão são definidos, todas as páginas que forem abertas na sequência podem ter acesso a essas informações (SILVA, 2014).

Subseção A – Cookies

O armazenamento dos cookies é feito no navegador remoto, a fim de facilitar o rastreamento e a identificação do usuário que está utilizando. Para criar um cookie, utilizamos a função setcookie, que necessariamente precisa receber seu nome e valor. Para o cookie pode ser definido um tempo de duração, que é opcional, e deve ser passado também por parâmetro na função de criação.

Como eles fazem parte do cabeçalho do HTTP, para que funcionem corretamente, a chamada setcookie deve ser feita antes que qualquer saída seja enviada ao navegador.

```
<?php
   //Criação do Cookie com duração de
60 segundos

setcookie("nome","Felipe",time()+60);
?>
```

Depois do cookie criado, ele é enviado para o navegador que o armazena. A cada chamada ao site que o usuário fizer, o cookie é enviado de volta ao servidor, podendo assim ser recuperado. Para acessar os valores do cookie utilizamos o array $_COOKIE;

```
<?php
   echo $_COOKIE["nome"];
   //Felipe
?>
```

Subseção B – Session

O funcionamento da sessão é parecido com o cookie, com a diferença que a sessão fica armazenada no servidor. As informações ficam visíveis apenas para o servidor, enquanto no cookie as informações são visíveis no navegador do usuário.

Em toda página que for necessária a utilização dos dados salvos na sessão, devemos fazer a chamada à função session_start(), que inicializa uma nova sessão ou resume uma sessão existente.

```
<?php
   session_start();
?>
```

Depois da sessão iniciada, podemos acessar os valores armazenados no array $_SESSION como se fossem um array tradicional.

```
<?php
   session_start();
   $_SESSION["nome"] = "Felipe";
   echo $_SESSION["nome"];
   //Felipe
?>
```

O exemplo acima produz o mesmo resultado do exemplo com cookie. Em ambos é armazenado o nome Felipe.

A utilização da sessão pode ser feita, por exemplo, quando há a necessidade de um login em uma página. Para que o usuário não tenha que ficar enviando o login e a senha a cada página acessada, ele faz isso apenas na primeira e, então, essas informações são salvas na sessão. Enquanto a sessão estiver iniciada, o servidor tem acesso a elas.

Para limpar a sessão, ou seja, retirar o conteúdo do array $_SESSION, utilizamos a função session_unset(), que libera todas as variáveis utilizadas na sessão. E pode-se também fazer uma chamada à função session_destroy() para encerrar a sessão. Porém, ele não apaga nenhuma variável global associada à sessão. Para poder utilizar as variáveis novamente, depois do session_destroy(), basta chamar o session_start().

Retomando a aula

Chegamos ao final da terceira aula. Vamos recordar?

1 – Obtendo informações do Usuário (GET e POST)

Nesta seção, foi tratada a maneira como o usuário interage com o servidor, enviando-lhe informações. A interação dos sistemas web fica normalmente a cargo dos formulários que podem submeter ao servidor via GET e POST os dados que forem inseridos nele.

2 – Funções e Escopo

Em seguida foi demonstrada a utilização de funções que facilitam a escrita de códigos que necessitem de muita repetição. Além disso, foi explicada a utilização de variáveis globais e locais delimitadas por seus respectivos escopos.

3 – Cookies e Sessions

Por fim, foram detalhados os cookies e sessões que tenham por função o armazenamento de informações no lado do usuário e no servidor, respectivamente. Essas informações podem ser muito úteis na criação de logins ou mesmo na personalização, quando um usuário específico é identificado.

Vale a pena

Vale a pena ler,

SOARES, W. Programação *Web com PHP 5*, Érica, 2014.

BEIGHLEY, L; MORRISON, M. *Use a Cabeça!* PHP & MySQL. Alta Books, 2010.

SILVA, J. M. C. *PHP na prática 200 Exercícios resolvidos.* Elsevier, 2014.

BENTO, E. J. *Desenvolvimento web com PHP e MySQL.* Editora Casa do Código, 2014.

Vale a pena **acessar**

PHP. In: PHP Documentation, 2019. Disponível em: https://www.php.net/docs.php. Acesso em: 20 out. 2019.

W3SCHOOLS. In: PHP Form Handling, 2019. Disponível em: https://www.w3schools.com/php/php_forms.asp. Acesso em: 22 out. 2019.

IMC. In: Cálculo IMC, 2019. Disponível em: https://www.programasaudefacil.com.br/calculadora-de-imc. Acesso em: 22 out. 2019.

Alura. In: HTTP: Diferenças entre GET e POST, 2019. Disponível em: https://www.alura.com.br/artigos/diferencas-entre-get-e-post. Acesso em: 22 out. 2019.

Minhas anotações