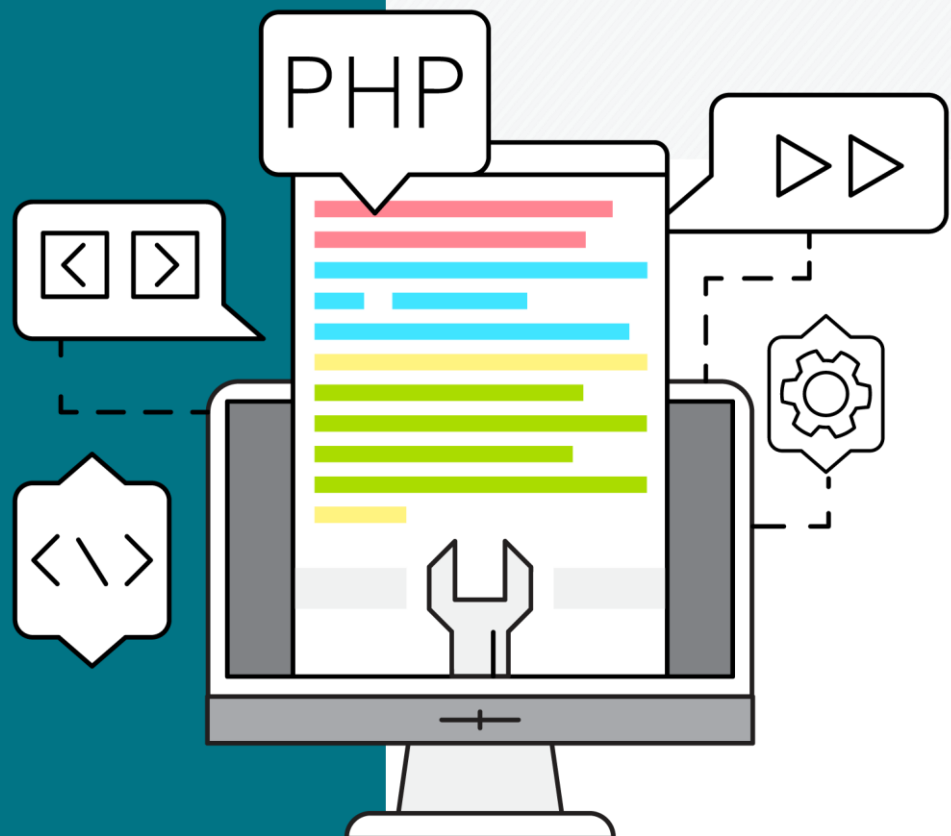


AGENDA 2

PHP: VARIÁVEIS E MÉTODOS GET E POST





Na linguagem PHP, também utilizamos variáveis, contudo, diferente de outras linguagens como C#, Java e C++ que precisamos indicar na declaração da variável um tipo (int, double, string), o PHP define o tipo de variável de forma dinâmica, em outras palavras, uma variável pode conter valores de diferentes tipos em diferentes momentos da execução dos comandos.

Então, basicamente, nas linguagens de programação citadas anteriormente, você em algum momento terá que declarar o tipo de dados que a variável recebe, essas linguagens são comumente chamadas de “linguagens fortemente tipadas”. No php a definição de um tipo em variável não existe, quando a variável recebe um valor, o PHP irá determinar automaticamente seu tipo em tempo de execução.

Alguns tipos de dados que a linguagem PHP suporta são:

- Inteiro
- Ponto flutuante
- String - Letras, números e caracteres especiais
- Booleanos

As variáveis no PHP são identificadas pelo caracter “\$” (cifrão). Então para criar uma variável basta colocar o cifrão seguido pelo nome da variável. O php é uma linguagem case-sensitive, ou seja, letras maiúsculas e minúsculas são diferentes, então os nomes das variáveis seguem as mesmas regras, como todos os outros códigos em PHP. Um nome de variável válido em php pode iniciar com uma letra ou underline(sublinhado), seguido de qualquer quantidade de letras, números ou sublinhados. Exemplos:

```
<?php
$nome = "Zeca da Silva";
$salario = 1000.50;
?>
```

Perceba que não indicamos em momento algum qual o tipo de variável, apenas atribuímos o valor na variável, e o php fica responsável pelo restante.

As variáveis numéricas não possuem nenhum mistério, sendo **Inteiro** - Um inteiro é qualquer número sem casas decimais, positivo ou negativo. **Ponto Flutuante** - Números de ponto flutuante, também conhecidos como "floats", "doubles" ou "números reais", números que apresentam fração. Exemplo: 1.0

Obs.: O php determina que o número deixa de ser inteiro para ser ponto flutuante no momento em há a necessidade de uso da casa decimal, seguindo essa ideia se colocarmos o valor 1.0 em uma variável o Php a tornará em uma variável de ponto flutuante.

Exemplo:

```
<?php
$valor = 1.0; // Variável depois da atribuição de valor passa a ser do tipo
float.
?>
```

Obs.: Outro fator muito importante que precisamos levar em consideração é saber que em PHP o separador de casas decimais também é o ponto, assim como em java.

String

Como já sabemos, trata-se do tipo que representa um conjunto de caracteres alfanuméricos (letras, números e caracteres especiais), portanto, é utilizado para armazenar e manipular textos. Para se atribuir valores a uma variável **String**, utilizam-se **aspas simples** ou **duplas**. Apesar de ambas as formas serem igualmente permitidas (aspas simples ou duplas), existem pequenas diferenças em seu uso.

Concatenação

Obs.: Para enviar à tela uma mensagem somando textos e conteúdos das variáveis, a concatenação é realizada por meio do **ponto final “.”**

Exemplo:

```
<?php
$nome = "Zeca da Silva";
$salario = 1000.50;
echo "Nome: ".$nome." Salario: ".$salario;
?>
```

Para entender melhor vamos programar!

No Visual Studio Code, crie um arquivo e o salve dentro da pasta Agenda 2 com o nome de “**desvendandoString**”, não esquecendo de escolher o tipo **PHP**. Veja a codificação deste exemplo:

```
<?php
$nome = "Zeca";
$email = 'zeca.silva@teste.com';

echo $nome."<br>";
echo $email."<br>";

?>
```

Após salvar e executar, perceberá que o resultado será como o representado na imagem 3, demonstrando que, nesse caso, o funcionamento de aspas simples ou duplas tem o mesmo resultado.

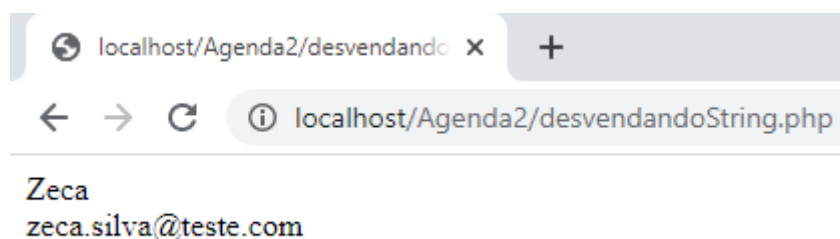


Imagem 4 – Resultado no navegador do arquivo desvendandoString.php.

Agora altere o código para:

```
<?php
$nome = "Zeca";
$email = 'zeca.silva@teste.com';

$texto1 = 'Olá $nome, seu email é: $email';
$texto2 = "Olá $nome, seu email é: $email";
echo$texto1."<br>";
echo$texto2."<br>";
?>
```

Após salvar e utilizar no navegador a mesma **URL** (<localhost/Agenda2/desvendandoString.php>), o resultado será como o representado na Imagem 5.

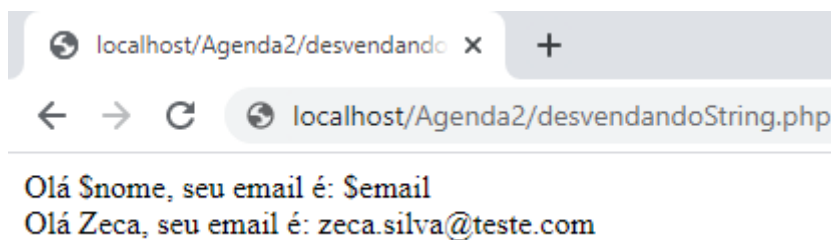


Imagem 5 – Resultado no navegador do arquivo `desvendandoString.php` após alterações.

Podemos notar que na primeira linha, na qual a string foi construída utilizando a aspa simples, o texto representado pelo navegador é literal:

“Olá \$nome, seu email é: \$email”

Mas quando são utilizadas aspas duplas para a construção da **string**, as variáveis retornam o valor dos seus conteúdos, deixando o resultado desta forma:

“Olá Zeca, seu email é: zeca.silva@teste.com”

Código HTML

Obs.: Nos exemplos utilizados para enviar a mensagem para o navegador podemos perceber que na concatenação do código `echo $nome."
";` estamos enviando uma codificação **HTML**, então o navegador recebe a informação e interpretará os códigos HTML normalmente, gerando assim uma quebra de linha.

Booleanos

O tipo de dados booleano representa um valor lógico binário, ou seja, tem apenas dois estados possíveis, que pode ser VERDADEIRO ou FALSO, portanto, esses tipos de variáveis são amplamente utilizados para verificação e atribuição de condições. Em PHP, os valores armazenados nas variáveis são verdadeiros (TRUE) e falso (FALSE) representados pelas palavras reservadas TRUE e FALSE.

Mas isso é comum em, basicamente, todas as linguagens, o que muda no PHP? Vamos lá! lembram da flexibilidade do PHP mencionada algumas vezes. O tipo booleano traz alguns exemplos:

- O valor 1 é considerado verdadeiro, enquanto o valor NULL é considerado falso.
- Valores numéricos diferentes de 0 (zero) são considerados como TRUE, enquanto o zero é considerado FALSE.
- Valores strings preenchidos são considerados TRUE, enquanto strings vazias ("") e o texto "0" é tido como FALSE.
- Arrays vazios ou objetos sem conteúdo são considerados como FALSE, já o inverso é logicamente TRUE.
- O valor NULL é considerado FALSE.



Utilizando o que foi visto até agora....

1. Crie um arquivo PHP na pasta Agenda 2.
 - a. Neste arquivo, crie 5 variáveis para armazenar respectivamente:
 - Nome Completo
 - Idade;
 - Profissão;
 - Salário.
 - b. Exiba, utilizando o Comando echo, os valores armazenados nas variáveis

A seguir, confira se você conseguiu resolver os desafios propostos!

```
<?php
$nomeCompleto = "Jéssica";
$idade = '17';
$profissao = 'Estagiária';
$salario = 750.00;

echo "Nome Completo: $nomeCompleto<br>";
echo "Idade: $idade<br>";
echo "Profissão: ". $profissao."<br>";
echo "Salário: ". $salario."<br>";

?>
```

Resultado no Navegador.

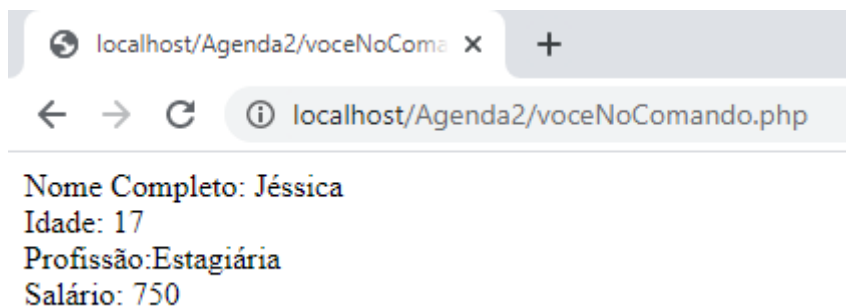


Imagem 6 – Resultado no navegador do exercício proposto.

Formulários, métodos GET e POST

Pense em formulários na web que você utiliza em seu cotidiano. Quando você preenche um formulário e clica no botão “**enviar**”, o formulário é submetido ao navegador e enviado para o servidor fazer esse processamento (**request**). Para que possamos trabalhar com as *requests* no servidor web, primeiro precisamos dessas requisições enviadas. O vídeo a seguir, explica de forma muito simplificada esse procedimento. Assista!



Fonte: Como funciona uma requisição HTTP? – (SPACE RAIL, 2020). Disponível em <https://www.youtube.com/watch?v=fhAXqcD21iE>. Acessado em 22/02/2020.

Para entender ainda mais, vamos programar!

No Visual Studio Code, crie dois arquivos e os salve dentro da pasta **Agenda 2**. O primeiro com o nome de “**request**” e o segundo “**acao**”, não se esquecendo de escolher o tipo **PHP** para ambos os arquivos.

No arquivo *request*, vamos desenvolver um formulário, com um campo de texto e um botão para enviar o conteúdo. A codificação deve ser desenvolvida desta forma:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>GET e POST</title>
</head>
<body>
<form id="form1" name="form1" method="get" action="acao.php">
<br>
    Nome <input name="nome" type="text" id="nome" placeholder = "Digitar
Nome">
<input name="Enviar" type="submit" id="enviar" value="enviar">
</form>
</body>
</html>
```


A tag **form** contém um atributo **action** com o valor “**acao**”. Com essa configuração o arquivo “**acao.php**” será executado assim que for realizada a ação do clique no formulário. No outro atributo “**method**”, com o valor atribuído **GET**, indica que os dados serão transmitidos diretamente pela **URL** da página. Este método possui algumas limitações, destacando-se:

- **URL** não pode passar de **255 caracteres**, portanto, se você estiver trabalhando com variáveis muito extensas, não é aconselhável o uso desse método.

Como dito, o método **GET** utiliza a **URL** do site para **enviar as requisições**. Há um caractere que **indica o início da criação das variáveis** e outro caractere que **faz a separação entre as variáveis**. Você precisa sempre estar atento ao conceito de que toda variável criada tem o seu valor atribuído, mesmo que esse valor seja nulo.

Para continuarmos, vamos programar o **arquivo ação**, com o seguinte código:

```
<?php
$nome = $_GET['nome'];
echo "Seja bem-vinda!". $nome;
?>
```

Perceba que é utilizada uma variável super global padrão “**\$_GET[]**” e, entre os colchetes e as aspas simples, está escrito nome. Este valor corresponde ao conteúdo atributo “**name**” do input de texto criado no formulário do arquivo “**request.php**”. Desta maneira, a variável “nome” terá o valor que for digitado pelo usuário no campo de texto.

Ao testar, o resultado é apresentado na imagem a seguir:

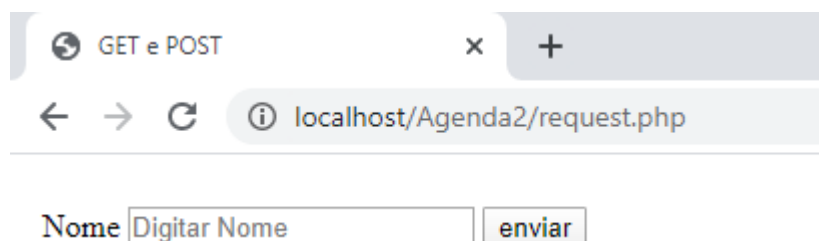


Imagem 7 –Resultado no navegador do arquivo request.php.

Após digitar um nome no campo de texto e clicar no botão enviar, o resultado deverá ser como o apresentado na imagem a seguir.

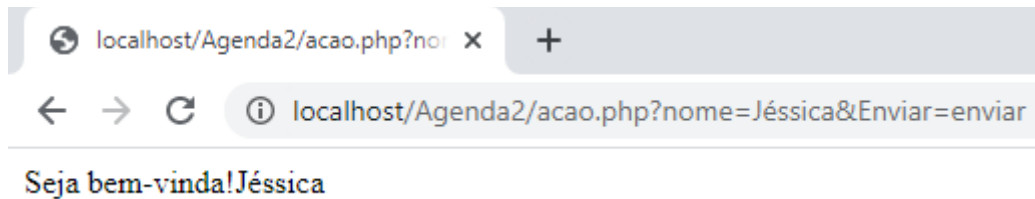


Imagem 8 – Resultado no navegador do arquivo acao.php. Apresenta a frase sejabem- vinda! Jéssica.

Perceba que na **URL** do seu navegador, após o nome do arquivo “**request.php**”, terá “**?nome=Jéssica**”, ou seja, os dados são transferidos, por meio da URL do próprio navegador, como é possível visualizar na imagem a seguir.

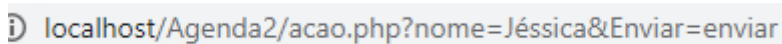


Imagem 9 – enviando dados via URL.

No método **POST**, a transferência de dados é realizada de forma oculta junto ao protocolo HTTP. Com este método teremos algumas vantagens como:

- Não há limite de tamanho dos dados que estão sendo enviados, ao contrário do que acontece com o método **GET**;
- Por meio do método **POST**, é possível enviar outros tipos de dados o que não é possível com o método get.

Para testar este método, basta alterar o arquivo “**request.php**”, alterando o valor “**get**” do atributo “**method**” pelo valor “**post**”, resultando no seguinte código:

```
<form id="form1" name="form1" method="post" action="acao.php">
```

Ao executar o código, você obterá o mesmo resultado, porém perceba pela imagem 9 que nenhuma informação será passada pela **URL**:

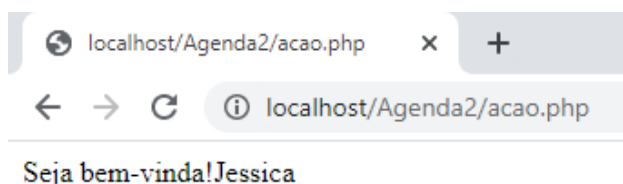


Imagem 10 – resultado do método Post



Utilizando o que foi visto até agora....

1 - Crie um arquivo PHP na pasta Agenda 2.

a. Neste arquivo, crie um formulário com os campos:

- Nome Completo
- Idade;
- Profissão;
- Salário.

2 - Crie um arquivo PHP para receber a ação do botão enviar.

- b. Este deverá exibir no navegador uma informação em cada linha.
- c. Utilize o método que você achar melhor.

A seguir, confira se você conseguiu resolver os desafios propostos!

Arquivo Formulário

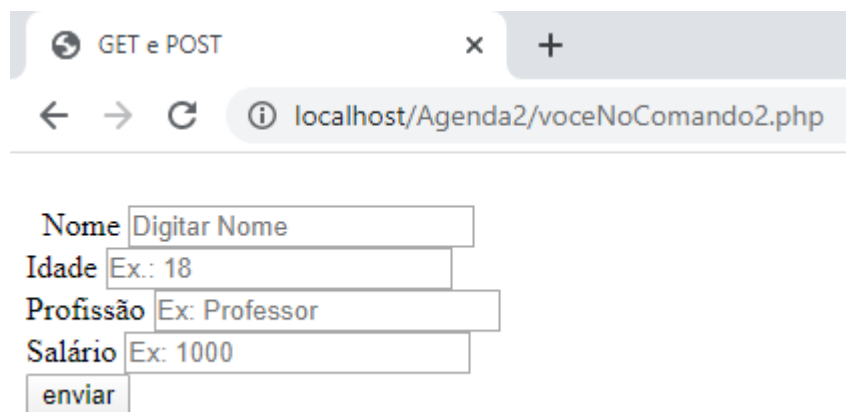
```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>GET e POST</title>
</head>
<body>
<form id="form1" name="form1" method="post" action="voceNoComando2A2.php">
<br>
    Nome <input name="nome" type="text" id="nome" placeholder = "Digitar
Nome"><br>
    Idade<input name="idade" type="text" id="idade" placeholder = "Ex.: 18"><br>
    Profissão<input name="profi" type="text" id="profi" placeholder = "Ex:
Professor"><br>
    Salário<input name="sal" type="text" id="sal" placeholder = "Ex: 1000"><br>
    <input name="Enviar" type="submit" id="enviar" value="enviar">
</form>
</body>
</html>
```

Arquivo de Ação

```
<?php
echo "Nome: " . $_POST['nome'] . "<br>";
echo "Idade: " . $_POST['idade'] . "<br>";
echo "Profissão: " . $_POST['profi'] . "<br>";
echo "Salário: R$ " . $_POST['sal'] . "<br>";
?>
```

Resultado no Navegador.

Arquivo Formulário



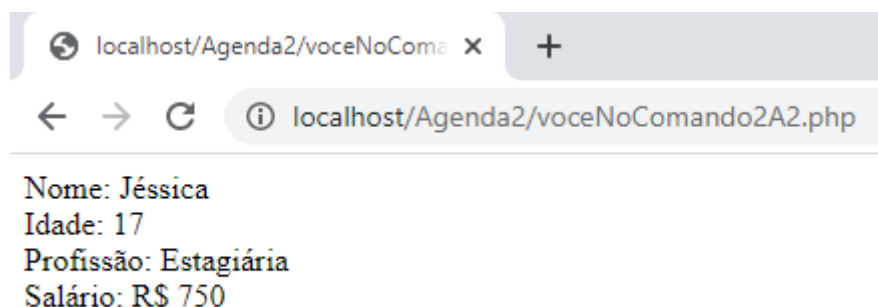
Nome

Idade

Profissão

Salário

Imagem 11 – Possível resultado no navegador do Exercício Você no Comando.



Nome: Jéssica
Idade: 17
Profissão: Estagiária
Salário: R\$ 750

Imagem 12 – Possível resultado no navegador do Exercício Você no Comando.