Programação para Internet

INSTITUTO FEDERAL
SANTA CATARINA
Câmpus Canoinhas

Formulários no Cliente

Vamos relembrar como criar um formulário em HTML. A tag **form** cria um formuláiro. Os dois atributos principais são **method** e **action**. O method diz respeito ao método HTTP que será utilizado. Iremos aprender a utilizar os métodos GET e POST. O atributo action é o endereço HTTP que a será requisitado quando o formulário for enviado.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
<h3>Cadastro de cliente</h3>
<form method="POST" action="form1_resp.php">
    Nome:
    <input name="nome" type="text" value="" /><br />
    Idade:
    <input name="idade" type="text" value="" /><br />
    <input type="submit" value="Cadastrar" />
</form>
</body>
</html>
```

Formulários no Cliente

Precisamos colocar campos no formulário, senão não terá informação para enviar. Para isso utilizamos a tag **input**. A seguir estão alguns atributos desta tag.

Atributo	Valor	Descrição
name	um identificador	É utilizado para recuperar o valor no servidor
value	valor inicial do campo	Quando carregar a página, irá mostrar este valor
type	text (padrão), button, checkbox, file, hidden, password, radio, reset, submit e image. Estes são HTML 4.	Tipo de campo

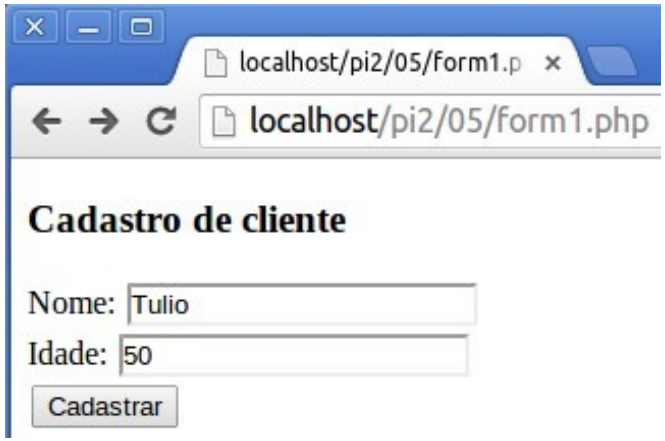
Formulários no Cliente

Observações:

- Normalmente, um formulário terá um campo do tipo submit. Este campo é um botão que envia o formulário.
- O tipo image é um botão que usará uma imagem invés de um botão do navegador.
- O tipo hidden não é renderizado, mas o cliente pode manipulálo no HTML.
- O tipo reset é um botão que volta os valores de todos os campos do formulário para o padrão.
- Para utilizar o tipo file, o formulário deve ter o atributo **enctype** com o valor **multipart/form data**.
- O HTML5 traz novos tipos de campos: color, date, datetime, datetimelocal, email, month, number, range, search, tel, time, url e week.

Formulários no Cliente

Formulários no Servidor



Como que eu faço para pegar os dados enviados pelo HTTP?

No PHP existem várias variáveis que já vem predefinidas na linguagem. Existem 3 variáveis superglobais que guardam os dados enviados pelo HTTP. Uma variável superglobal está acessível em qualquer escopo! Essas variáveis normalmente possuem o prefixo **$_** e seu nome usa caixa alta. Exemplo: $_POST.

Formulários no Servidor

Para recuperar os dados do HTTP, podemos utilizar:

- $_GET: guarda os dados quando o HTTP usa o método GET.

  $_POST: guarda os dados quando o HTTP usa o método POST.

  ~~$_REQUEST~~: guarda os dados do GET, do POST e do COOKIE. **Não utilizar, pois reduz a segurança da aplicação.**

  $_GLOBALS

Essas superglobais são arrays associativos. A chave é o atributo *name* do campo HTML.
Exemplo: o campo idade tinha o atributo *name* valendo "idade". Portanto para acessar o valo enviado, podemos utilizar $_POST['idade'].

Essas variáveis podem ser alteradas, porém isso não deve ser feito, pois podem deixar o código confuso.

Formulários no Servidor

Vamos ver como que a página de resposta foi escrita.

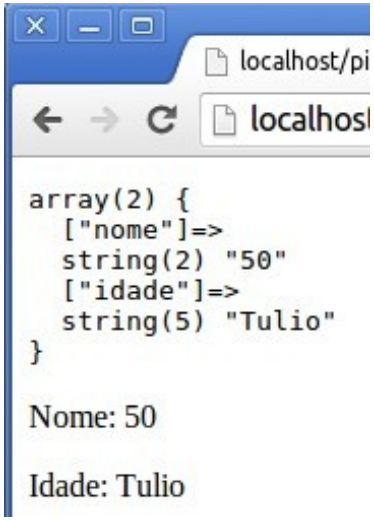
```php
<pre>
<?php var_dump($_POST); ?>
</pre>
<?php
echo "<p>Nome: $_POST[nome]</p>";
echo "<p>Idade: {$_POST[idade]}</p>";
//echo 'OK: ' . $_POST['nome'];
//echo "<p>Erro: $_POST['erro no parse']</p>";
?>
```

Neste caso o nosso código mostra o que tem na variável $_POST. Perceba que idade também é string. Todo dado que vem do cliente é string.

Também podemos notar que se imprimimos um array dentro de uma string (aspas duplas), não devemos colocar aspas na chave.

Formulários no Servidor

Vamos parar para pensar. O cliente pode enviar qualquer coisa como dado, certo? Então, no campo idade, ele poderia enviar o nome e no campo nome poderia enviar a idade!



Agora você aprenderá uma regra de ouro! Tudo que o cliente enviar deve ser validado.

Exercícios

Crie 2 páginas PHP. Uma com um formulário contendo: nome, email e mensagem. A outra página deve exibir os dados enviados. O nome deve aparecer em cima. Do lado do nome e entre parênteses deve estar o email. Em baixo do nome deve ser exibida a mensagem. Utilizem o método HTTP mais adequado no formulário.

- Tag form: atributos method e action.
- Tag input: atributos name, type (text, submit) e value. Supreglobal PHP do método POST: $_POST.

Enviando dados por GET

Antes de entrarmos em validação, vamos ver como enviar dados utilizando GET. O método GET do HTTP envia os dados pela **URL**. A URL a seguir é de uma página qualquer.

```
localhost/pagina.php
```

Quando queremos passar dados por GET, colocamos um "?". Isso se chama **query**.

```
localhost/pagina.php?
```

Depois do "?", podemos colocar os dados no formato "chave=valor".

```
localhost/pagina.php?nome=Daniel
```

Alguns caracteres especiais serão codificados na URL. Exemplo: espaço é %20.

Enviando dados por GET

Mas como fazemos para enviar vários dados por GET? Neste caso usamos o "&" como separador.

```
localhost/pagina.php?nome=Daniel&idade=13
```

Devemos ter alguns cuidados com o GET:

- Ele normalmente **não** é utilizado em formulários. O usuário verá os dados na URL.
- A URL pode ser gravada nos favoritos ou copiada.
- A URL possui um tamanho máximo! Até 2000 caracteres a URL provavelmente funcionará
- com qualquer software. Mesmo assim, uma URL tão grande pode parecer um projeto mal feito. URLs grandes, hoje, possuem cerca de 150 caracteres.

Enviando dados por GET

Agora nós sabemos como os dados são enviados pelo GET. Mas como fazemos uma requisição GET?

- Na barra de endereço do navegador. Em um hyperlink na página.
- Em um formulário com *method* valendo GET ou POST.
-

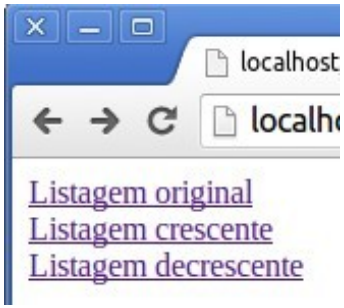
Se o formulário usar o POST, então os campos serão enviados no corpo da mensagem e a URL da action será utilizada sem alteração. A URL pode ter campos GET. O servidor sabe que a requisição é POST, porém mesmo assim ele recupera os campos da URL.

Se o formulário usar o GET, então os campos serão enviados na URL. O navegador irá descartar quaisquer outros campos que já estiver na URL.

Enviando dados por GET

Observe a página a seguir. Ela possui 3 hyperlinks. Todos vão para a mesma página. A diferença entre eles está na query. O primeiro não possui. O segundo possui o parâmetro ordem valendo "crescente" e o último possui o parâmetro ordem valendo "decrescente".

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
</head>
<body>
<a href="get_resp.php">Listagem original</a><br />
<a href="get_resp.php?ordem=crescente">Listagem crescente</a><br />
<a href="get_resp.php?ordem=decrescente">Listagem decrescente</a>
</body>
</html>
```



Enviando dados por GET

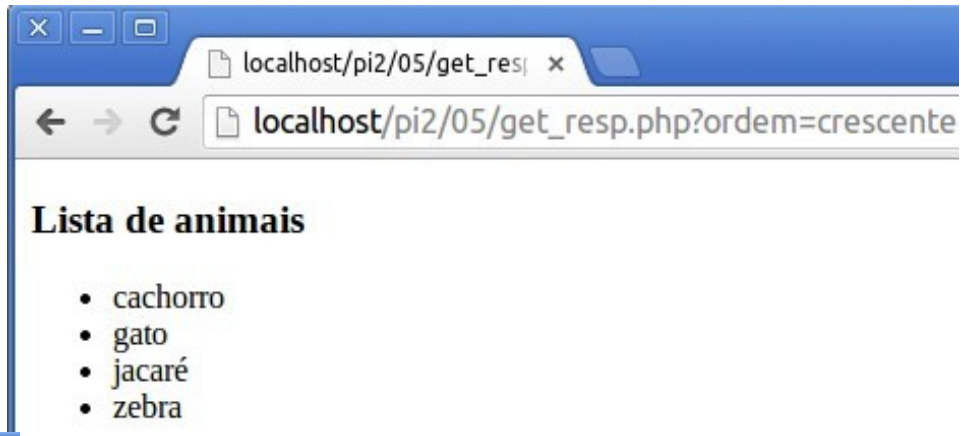
Vamos dar uma olhada na página "get_resp.php". Ela usa a superglobal $_GET para acessar os dados da URL. Primeiramente existe um array de animais. Depois é checado o valor do parâmetro GET. Dependendo do valor, o array de animais é ordenado.

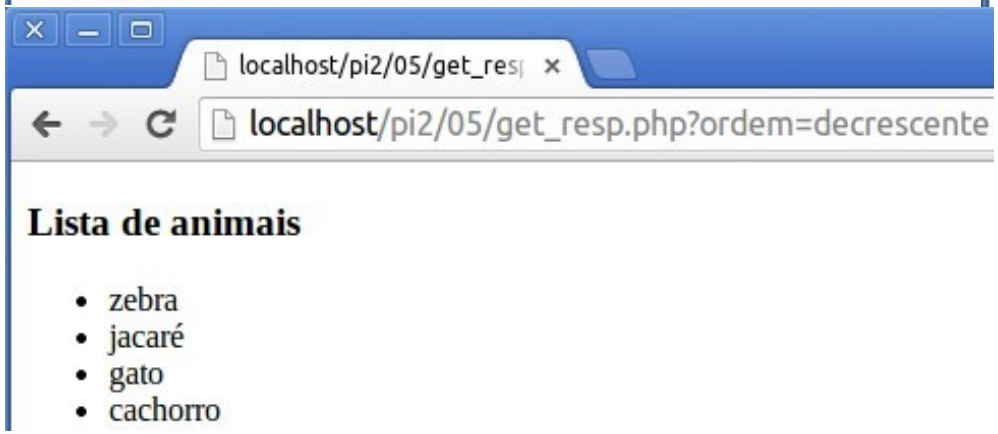
```php
<?php
$animais = array('jacaré','gato','cachorro','zebra');
if ($_GET['ordem'] == 'crescente') {
    sort($animais);
} elseif ($_GET['ordem'] == 'decrescente') {
    rsort($animais);
}
?>
<!DOCTYPE html>
<html>
<head><meta charset="UTF-8" /></head>
<body>
<h3>Lista de animais</h3>
<ul>
<? foreach ($animais as $animal) : ?>
    <li><?= $animal ?></li>
<? endforeach; ?>
</ul>
</body>
</html>
```

Perceba que foi utilizada uma notação alternativa para o *foreach*. Poderia ser utilizada a notação comum (que utiliza chaves). Assim como o foreach, as outras estruturas também possuem uma notação alternativa.

Enviando dados por GET

A primeira, segunda e terceira imagem correspondem ao primeiro, segundo e terceiro hyperlink respectivamente.





Exercícios

Crie uma página PHP com uma tabela de computadores. A tabela deve ter as seguintes colunas: CPU, placa de vídeo e memória. Em baixo da tabela devem ter 2 hyperlinks. Eles servem para ordenar a tabela. A tabela é ordenada pela coluna CPU em ordem crescente ou decrescente.

GET ou POST?

Como eu sei se devo utilizar GET ou POST? A seguir algumas regras:

- Se a requisição alterar um dado no servidor, então deve ser POST. Exemplo: cadastrar, editar ou excluir.
- Se a requisição tiver dados sigilosos, então deve ser POST. Exemplo: senhas.
- Normalmente um formulário utiliza POST. Os outros casos provavelmente serão GET.
-

Referências

NIEDERAUER, Juliano. **Desenvolvendo websites com PHP**. 2. ed. São Paulo: Novatec, 2011. 301 p. ISBN 9788575222348.