

SI401

Programação para a Web

2º Semestre - 2024

Programação *Back-End*: PHP

Unidade 15

Prof. Guilherme Palermo Coelho
guilherme@ft.unicamp.br

Roteiro

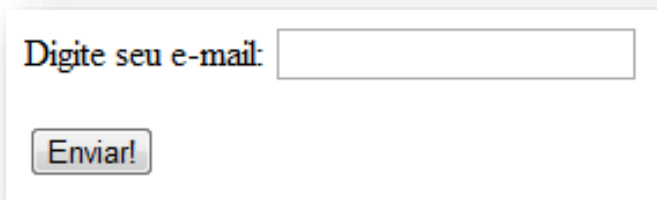
- PHP e formulários HTML;
- *Server Side Includes*;
- Referências.

PHP E FORMULÁRIOS HTML

PHP e Formulários HTML

- A principal forma de interação entre programas PHP e usuários é através dos *formulários HTML*;
- Estes formulários (criados com a *tag* **<form>**) podem conter diversos *elementos de entrada de dados*;

```
<form>
  <p>Digite seu e-mail: <input type="text" name="eml" size="20" /> </p>
  <p><input type="submit" value="Enviar!" name="enviar" /></p>
</form>
```



Digite seu e-mail:

PHP e Formulários HTML

- No exemplo anterior, os dados são ***perdidos*** ao se clicar no botão “Enviar!”;
 - O navegador não sabe para onde enviar os dados do formulário;
- Para que o exemplo anterior se torne útil, é preciso indicar ao navegador o ***destino*** para o qual os dados de um formulário devem ser enviados:
 - Neste curso, este destino serão ***scripts em PHP***.
 - Isto é feito através da atribuição do caminho e nome do *script* PHP (arquivo .php) ao atributo ***action*** do elemento **`<form>`**.

Métodos de Envio de Dados

- Além de definir o programa PHP que receberá os dados no atributo ***action*** do formulário, é preciso definir também como estes dados serão passados;
- Existem dois métodos de passagem de parâmetros: **GET** e **POST**
 - Estes métodos são definidos no atributo ***method*** do elemento ***form***:
 - `<form action="recebe_dados.php" method="POST">`
 - `<form action="recebe_dados.php" method="GET">`

Métodos de Envio de Dados

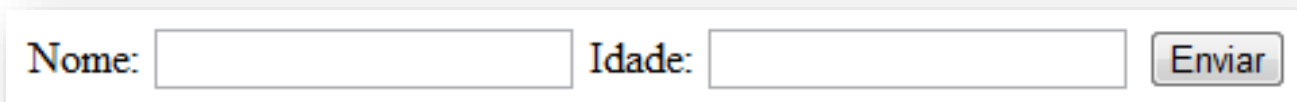
- Método GET

- O método GET é o método padrão para envio de dados;
 - Se nenhum método for especificado em *method*, GET é utilizado.
- Neste método, os dados são enviados *juntamente com o nome da página* (na URL) que processará os dados recebidos:

Métodos de Envio de Dados

- Método GET

```
<form action="teste2.php">
    Nome: <input type="text" name="fname" />
    Idade: <input type="text" name="age" />
    <input type="submit" />
</form>
```



- Se preenchermos os campos acima com “**Maria**” e “**23**”, a URL que será chamada ao pressionarmos o botão “**Enviar**” será:

<http://localhost:8080/teste2.php?fname=Maria&age=23>

Métodos de Envio de Dados

- Método GET

- Os campos do formulário serão passados como **parâmetros** após o endereço de destino;
 - O caractere “?” sinaliza o **início** de uma cadeia de variáveis;
 - O caractere “&” é um **separador** de variáveis;
 - As **variáveis** (nomes) e seus respectivos **valores** são separados por “=”;

<http://localhost:8080/teste2.php?fname=Maria&age=23>

Métodos de Envio de Dados

- **Método GET**

- A utilização do método GET possui **duas *desvantagens*** principais:
 - Os valores de todos os campos são visíveis para o usuário,
 - Isto pode ser um problema, no caso de passagem de senhas;
 - Existe um limite máximo de caracteres que podem ser enviados (cerca de 2.000);

Métodos de Envio de Dados

- **Método GET**

- GET também tem ***vantagens***:
 - Pode ser usado para passar ***parâmetros*** diretamente através do link;
 - **Ex.:** identificar um produto em uma loja virtual
- http://www.loja.com/produto.php?id_produto=23
- Permite que o usuário salve, nos favoritos, links com os valores já preenchidos de um formulário.

Métodos de Envio de Dados

- Método POST

- Para utilização do método POST, deve-se **obrigatoriamente** utilizar o atributo **method** da tag **<form>**:

```
<form action="teste2.php" method="POST">  
    Nome: <input type="text" name="fname" />  
    Idade: <input type="text" name="age" />  
    <input type="submit" />  
</form>
```

- Diferentemente do método GET, o método POST envia os dados do formulário por meio do **corpo da mensagem HTTP** enviada ao servidor;
 - O usuário não vê mais os valores dos parâmetros na URL, apenas o endereço do programa ativado.

Métodos de Envio de Dados

- Método POST

- Outra vantagem do método POST é que ***não há limitação*** de tamanho dos dados que estão sendo enviados.
- O método POST também permite o envio de ***outros tipos*** de dados que não podem ser enviados pelo GET, como, por exemplo, imagens ou outros arquivos;
 - Para isso, usa-se o valor ***file*** no atributo ***type*** do elemento **<input>**.
 - Mais informações sobre o envio de arquivos para *scripts* PHP:

<http://www.php.net/manual/en/features.file-upload.php>

Tratando informações recebidas

- Depois de especificar qual programa PHP receberá os dados de um formulário e como estes dados serão enviados, resta saber **como** trabalhar estes dados;
- Para acessar estes dados, basta utilizar dois **arrays predefinidos** pelo PHP:
 - O array `$_GET` (usado para valores passados pelo método GET);
 - O array `$_POST` (usado para valores passados pelo método POST);
- Em ambos os casos, os **nomes** dos campos no formulário HTML são usados como **chaves associativas**:
 - Os valores dos campos do formulário são armazenados como valores dos *arrays*;

Tratando informações recebidas

```
<form action="teste2.php" method="POST">
    Nome: <input type="text" name="fname" />
    Idade: <input type="text" name="age" />
    <input type="submit" />
</form>
```

- No programa ***“teste2.php”*** poderíamos acessar os valores preenchidos nos campos do formulário através de:
 - `$_POST[“fname”]` (valor do campo “Nome”);
 - `$_POST[“age”]` (valor do campo “Idade”);
- Se o método de envio tivesse sido GET, bastaria acessarmos:
 - `$_GET[“fname”]` (valor do campo “Nome”);
 - `$_GET[“age”]` (valor do campo “Idade”);

Funções especiais para formatação de dados

- Quando um programa em PHP recebe um conjunto de dados de um formulário, muitas vezes é necessário ***pré-processar*** estes dados antes de sua utilização;
- Para realizar este pré-processamento, PHP disponibiliza algumas funções especiais, que serão tratadas a seguir.
- **Função *htmlspecialchars(<string>)*:**
 - Esta função é útil para evitar que *strings* passadas pelo usuário, via formulários, contendo códigos HTML maliciosos sejam interpretadas pelo navegador como códigos HTML;
 - Por exemplo, inserção de imagens inapropriadas em um fórum de discussões.

Funções especiais para formatação de dados

- Função *htmlspecialchars(<string>)*:
 - Esta função recebe uma *string* como entrada e substitui todas as *tags* HTML por caracteres especiais:
 - & é substituído por &
 - " é substituído por "
 - < é substituído por <
 - > é substituído por >
 - Ao fazer estas substituições, o navegador não mais interpreta a *string* como código HTML, e sim como texto puro.

Funções especiais para formatação de dados

- Função *stripslashes*(<string>):
 - Quando o usuário digita em um formulário dados que contêm **caracteres especiais**, o PHP insere nestes dados o caractere de controle \ antes de cada caractere especial:
 - Ex.: “**Manolo**” da Silva seria recebido por um programa PHP como \“**Manolo**\” da Silva;
 - Para situações em que é necessário manipular estas *strings* sem os caracteres de controle, utiliza-se a função *stripslashes*;
 - Ex.: `stripslashes(“\“Manolo\” da Silva”)` gera a *string* “**Manolo**” da Silva.

Funções especiais para formatação de dados

- Funções *urldecode(<string>)* e *urlencode(<string>)*:
 - Quando o método de envio dos dados é GET, **todos** os caracteres que não são alfanuméricos ou *underscore* (_) são convertidos para um código hexadecimal precedido de %:
 - Ex.: “Maria da Silva” é enviada, via GET, pela seguinte URL:
<http://www.site.com/programa.php?nome=Maria%20da%20Silva>
 - Para converter estes caracteres de volta para texto normal, utiliza-se a função *urldecode*:
 - Ex.: \$nome = urldecode(“Maria%20da%20Silva”);
 - Neste exemplo, os códigos hexadecimais são convertidos de volta para os caracteres que eles representam.

Funções especiais para formatação de dados

- Funções *urldecode(<string>)* e *urlencode(<string>)*:
 - De maneira análoga, caso seja necessário enviar alguma *string* para outro programa PHP via URLs, os caracteres não alfanuméricos (ou *underscore*) precisariam ser ***codificados antes do envio***;
 - Isso pode ser feito pela função ***urlencode(<string>)***;
 - Esta função tem comportamento exatamente oposto a ***urlencode***, ou seja, substitui os caracteres não alfanuméricos pela respectiva codificação hexadecimal precedida de %.

SERVER SIDE INCLUDES (SSI)

Server Side Includes (SSI)

- É possível inserir o **conteúdo** de um arquivo PHP **dentro de outro antes** que o servidor execute o programa PHP;
- Esta possibilidade simplifica várias atividades relacionadas a programação web:
 - Facilita a manutenção de partes de documentos HTML que são comuns a todos os documentos de um site:
 - Cabeçalhos e rodapés;
 - Menus, ...
 - Permite uma melhor organização de programas PHP e a reutilização de código.

Server Side Includes (SSI)

- SSI pode ser feito em PHP com as funções ***include()*** e ***require()***:
 - O funcionamento destas duas funções é idêntico, exceto pela forma com que cada uma trata erros:
 - ***include*** apenas emite um *warning* e permite que a execução do programa continue;
 - ***require*** emite um *erro* e a execução do programa é encerrada;
- Sintaxe:
 - ***include*** “caminho_e_nome_do_arquivo”;
 - ***require*** “caminho_e_nome_do_arquivo”;

Server Side Includes (SSI)

// Arquivo "menu.php"

```
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
```

// Arquivo "teste.php"

```
<html>

    <body>

        <div class="sitemenu">
            <?php include "menu.php"; ?>
        </div>
        <h1>Bem vindo à página de teste!</h1>
        <p>Bla bla bla.</p>
    </body>
</html>
```


Server Side Includes (SSI)

```
// Arquivo resultante para o navegador
<html>
  <body>
    <div class="sitemenu">
      <a href="/default.php">Home</a>
      <a href="/tutorials.php">Tutorials</a>
      <a href="/references.php">References</a>
      <a href="/examples.php">Examples</a>
      <a href="/about.php">About Us</a>
      <a href="/contact.php">Contact Us</a>
    </div>
    <h1>Bem vindo à página de teste!</h1>
    <p>Bla bla bla.</p>
  </body>
</html>
```

Server Side Includes (SSI)

- No caso de utilização de SSI para reaproveitamento de código, recomenda-se a utilização da função ***require***;
- Um script não deve ser executado se todas as funções não estiverem devidamente carregadas.

EXERCÍCIO SUGERIDO 2

Exercício Sugerido 2

- Crie um documento HTML com os seguintes campos para cadastro de uma pessoa: ***nome completo, CPF, RG, idade, sexo, estado civil, telefone fixo, telefone celular e endereço***. Quando estes campos forem preenchidos e o usuário clicar no botão “***Cadastrar***”, os dados devem ser enviados para um script PHP que exibirá o cadastro feito pelo usuário em um novo documento HTML. Garanta que este novo documento gerado pelo *script* PHP siga a especificação HTML 5 (faça a validação no site do W3C).

REFERÊNCIAS

Referências

[1] Niederauer, J. “Desenvolvendo Websites com PHP”, 2ª ed. Novatec, 2011.

[2] W3Schools PHP Tutorial:

<http://w3schools.com/php/default.asp>

[3] Manual do PHP:

http://php.net/manual/pt_BR/index.php

[4] Hudson, P. “Hacking with PHP”:

<http://www.hackingwithphp.com/>

[5] Prettyman, S. "Learn PHP 7", Apress, 2016:

<https://link.springer.com/book/10.1007/978-1-4842-1730-6>