Mini-curso de introdução à Linguagem PHP

PHP: Hypertext Preprocessor

Por que aprender PHP ?

- PHP é uma linguagem simples, rápida (escalável) e poderosa.
- Mesmo sendo simples, pode ser utilizada para desenvolver sistemas robustos: ERPs, CRMs, e-coomerce, etc.
  - Basta comprometimento e organização da equipe desenvolvedora.

Por que aprender PHP ?

- Hoje em dia já suporta quase todos os paradigmas da POO.
- Possui uma documentação muito boa, com exemplos e comentários. Traduzida para vários idiomas, inclusive português:
  - http://php.net/manual/pt_BR/
- Possui *frameworks* consolidados (Zend2, Cake, CodeIgniter, Symfony2, Yii, Laravel, Phalcon, etc).

Por que aprender PHP ?

- Possui engines de “templatezação”: Smarty, Dwoo, Plates, Blade, etc
- E também de ORM: Doctrine, RedBean, Propel, etc
- E ainda é uma das linguagens *server-side* para desenvolvimento de web sites e sistemas web mais utilizadas:

Ranking

1) JavaScript
2) Java
3) PHP
4) C#
5) Python
6) C++
7) Ruby
8) C
9) Objective-C
10) CSS
11) Perl
12) Shell
13) Scala
14) Haskell
15) R
16) Matlab
17) Clojure
18) CoffeeScript
19) Visual Basic
20) Groovy

*Ranking divulgado em **Janeiro de 2014** pela empresa RedMonk com base nos projetos Git hub e das discussões no stackoverflow

Ranking

1) JavaScript
2) Java
3) PHP
4) Python
5) C#
6) C++
7) Ruby
8) CSS
9) C
10) Objective-C
11) Perl
12) Shell
13) R
14) Scala
15) Haskell
16) Matlab
17) Go
18) Visual Basic
19) Clojure
20) Groovy

*Ranking divulgado em **Janeiro de 2015** pela empresa RedMonk com base nos projetos Git hub e das discussões no stackoverflow

Ranking

1) JavaScript
2) Java
3) PHP
4) Python
5) C#
6) C++
7) Ruby
8) CSS
9) C
10) Objective-C
11) Shell
12) Perl
13) R
14) Scala
15) Go
16) Haskell
17) Swift
18) Matlab
19) Clojure
20) Groovy

*Ranking divulgado em **Janeiro de 2016** pela empresa RedMonk com base nos projetos Git hub e das discussões no stackoverflow

Ranking

1) JavaScript
2) Java
3) Python
4) PHP
5) C#
6) C++
7) CSS
8) Ruby
9) C
10) Objective-C
11) Swift
12) Shell
13) Scala
14) R
15) Go
16) Perl
17) TypeScript
18) PowerShell
19) Haskell
20) CoffeeScript

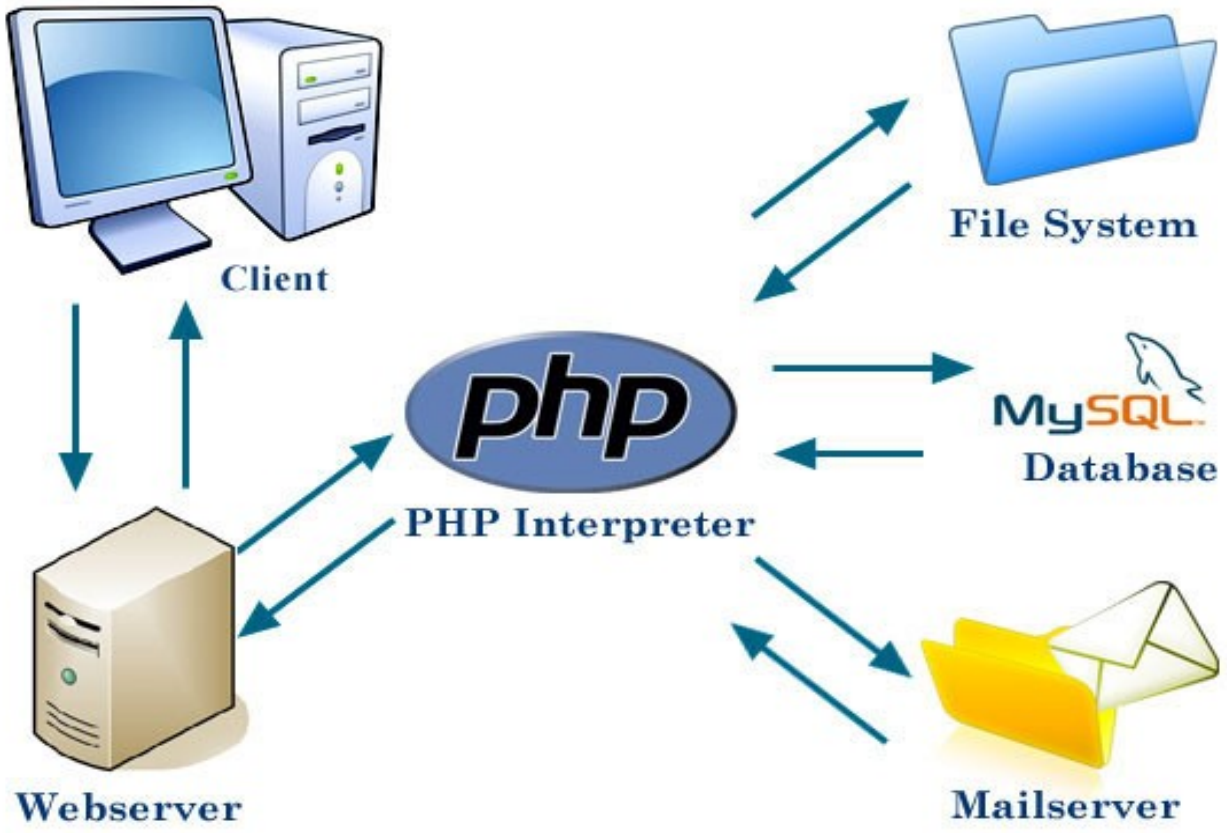
*Ranking divulgado em **Junho de 2017** pela empresa RedMonk com base nos projetos Git hub e das discussões no stackoverflow

Ranking

No ranking divulgado esse ano pela **IEEE**, a linguagem **PHP ficou em 8º lugar de 48 linguagens** analisadas. O ranking leva em consideração 12 métricas as quais são atribuídos pesos para cada uma delas:
http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

Arquitetura Cliente-Servidor

- Apesar de possuir Interface por Linha de Comando (CLI) e suporte à APIs de Interface Gráfica (GUI) como GTK e Qt, a maioria das aplicações PHP rodam na seguinte arquitetura:



- Um cliente (geralmente navegador) faz uma requisição ao servidor (WebSerber)
- O servidor encaminha a requisição para o módulo PHP (no caso do windows o php.exe)
- Assim, o PHP “pode acessar” todos os recursos do servidor: sistema de arquivos, bases de dados, servidor de e-mails, etc
- A requisição é processada e a resposta é devolvida ao Servidor Web, que por sua vez devolve ao cliente

Imagem retirada de 2n2media
http://www.2n2media.com/resource/mobile-application-development-using-php

Algumas Características

- PHP é uma linguagem interpretada, de *script, server-side* (roda do lado do servidor, retornando o resultado para o cliente: geralmente um navegador).
  - Ao mesmo tempo que é considerada uma linguagem simples e flexível, é também poderosa e capaz de produzir aplicações robustas.
  - Fracamente tipada com tipagem dinâmica.
  - Suporta a maioria dos paradigmas da Programação Orientada a Objetos.

Configurações

- As principais configurações da Linguagem PHP encontram-se no arquivo php.ini, configurações como:
  - Tempo máximo de execução do script
  - Cookies / Sessões
  - Região / Data e Hora
  - Bibliotecas / APIs externas
  - Exibição de erros
  - Sintaxes
  - etc

Configurações

- Em geral, após alterar uma configuração, o serviço referente ao servidor web (no nosso caso o Apache) deve ser reiniciado para ter efeito.
- Pode-se permitir que as configurações sejam alteradas em tempo de execução:
  - Pela função genérica:
    - `ini_set("parametro","valor");`
  - Ou por funções específicas:
    - `error_reporting(TIPO_ERRO);`

Tipos de Erros

- O PHP dispara diferentes tipos de erros, e a maioria não interrompe a execução do *script:*
  - **FATAL ERROR:** erro fatal que **interrompe** a execução do *script*.
  - **WARNING:** erro que acontece em tempo de execução mas **não** interrompe a execução do *script.*
  - **NOTICE:** aviso em tempo de execução de alguma inconsistência no código. Também **não** interrompe a execução do *script*.

Arquivos PHP

- Tradicionalmente os servidores são configurados para **executar** os **arquivos PHP** com a **extensão `.php`** (porém é possível alterar essa configuração ou mesmo 'esconder' a extensão)
- Da mesma forma, usualmente o código a ser executado pelo interpretador PHP é colocado entre as *tags* *<?php ... ?>*

Exemplo

- Seja um arquivo `PrimeiraAula.php`:

```php
<?php
    echo "Hello World !!!";
?>
```

- Abrindo o arquivo no navegador teremos:

Característica importante

- Uma característica importante é que “não importa” o que está fora das *tags* que delimitam o código PHP, e isso é importante quando programas para web para adicionarmos trechos de código HTML, CSS, JavaScript, etc.
- Conformes vimos no slide anterior, uma das funções utilizadas para “imprimir” *(output)* conteúdo no PHP é a função **`echo`**. E podemos imprimir qualquer coisa importante/útil, para nossa página, como por exemplo, HTML, CSS, JavaScript, etc

Exemplo

```
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <title>Primeira Aula PHP</title>
        <meta charset="utf-8">
        <style>
            body p {
                font-family: Verdana, sans-serif;
                font-size: 11px;
                padding: 10px;
                border: 1px solid #000;
            }
        </style>
    </head>
    <body>
        <?php
            echo "<p>Hello World !!!<p>";
        ?>
    </body>
</html>
```

Fracamente tipada? Tipagem dinâmica?

- Não é necessário declarar o tipo das variáveis (ou mesmo o retorno das funções) e “a qualquer momento”, pode alterar o tipo das variáveis, exemplo:

```php
<?php

    $a = "Olá";
    $a = 123456;
    $a = true;
    $a = 1.75;
    echo $a;

?>
```

No exemplo, a variável **$a** teve seu valor alterado entre tipo *string*, inteiro, booleano e ponto flutuante. O exemplo funciona normalmente em PHP sem retornar nenhum erro.

Comentários

- Comentários em PHP são idênticos a comentários em C:

```php
<?php

    //comentário em linha única
    echo "Hello World";
    /*
     * Comentário
     * de
     * múltiplas
     * linhas
     */

?>
```

Importante

- Por ser fracamente tipada, em uma **comparação booleana**, **todos** os casos abaixo são considerados **falsos**:

```php
$a=0;
$a="";
$a=false;
$a=null;

if ($a==false) {
    echo "<br />Variável <strong>a</strong> é falsa";
}
//ou ainda
if (!$a) {
    echo "<br />Variável <strong>a</strong> é falsa";
}
```

Importante

- Por sua vez, **todos** os casos abaixo são considerados como verdadeiro:

```php
$a=1;
$a="a";
$a=true;

if ($a==true) {
    echo "<br />Variável <strong>a</strong> é verdadeira";
}
//ou ainda
if ($a) {
    echo "<br />Variável <strong>a</strong> é verdadeira";
}
//em resumo, qualquer variável não vazia ou explicitamente false
//é considerada como verdadeira
```

O que fazer?

- Casso precisamos ter certeza que uma variável tenha um **determinado tipo e valor** devemos utilizar o **operador ===**

```php
$a=0;
$a="";
$a=false;
$a=null;

if ($a===false) {
    echo "<br />Variável <strong>a</strong> é falsa";
}
//apenas o caso em que $a é false entrará no condição
```

Em resumo:

- Operador “Igual” **==**
  - mesmo **valor**
- Operador “Idêntico” **===**
  - mesmo **valor** e **tipo**

Mais informações em:
http://php.net/manual/pt_BR/language.operators.comparison.php

Tendo um mínimo de controle

- Apesar de tudo a linguagem PHP ainda nos fornece um mínimo de controle sobre o tipo das variáveis
  - Por meio de *casting*, ex: `$a = (int) "157";`
  - E testando o tipo das variáveis, ex:
    ```
    if (is_int($a)) { ... }
    ```

Mais exemplos

```php
$a = ".75";
if (is_float($a)) {
    echo "<br />A variável <strong>a</strong> é ponto flutuante";
}
else {
    echo "<br />A variável <strong>a</strong> não é ponto flutuante";
    $a = (float) $a;
    if (is_float($a)) {
        echo "<br />Agora sim a variável <strong>a</strong> é ponto
flutuante: ".$a;
    }
}
```

O que o exemplo acima irá imprimir ?

Outras funções

- Outros exemplos de funções para verificação de tipo de variáveis:

```
is_bool($var)
is_float($var)
is_numeric($var)
is_string($var)
is_array($var)
is_object($var)
```

Ainda sobre tipos e valores

- Também há funções para a realização de outras checagens referentes a variáveis:
  - Se o valor é **nulo**:
    - `is_null($var);`
  - Se a variável foi **inicializada**:
    - `isset(var);`
  - Se a variável está **vazia**:
    - `empty(var);`

empty($var)

- Valores de variáveis que são considerados “vazio” pela função `empty(`*`$var`*`)` :

```php
$a = ""; //uma string vazia
$a =  0; //0 como inteiro
$a = "0"; //0 como string
$a = NULL;
$a = FALSE;
$a = array();
```

Concatenação

- A concatenação em PHP é feita utilizando o operador **. (ponto):**

```php
<?php

    //exemplo de concatenação
    $a = "conteúdo";
    $b = 111213;
    echo "Imprimindo a variável a: ".$a." e também a variável b ".$b;

?>
```

Arrays, vetores, etc

- Em PHP, arrays, vetores, listas e demais coleções são implementados utilizando *hash map* (chave → valor)
- Aliado a outras características da linguagem em relação a tipagem, faz com que arrays em PHP sejam extremamente flexíveis

Criando um array

- Assim como variáveis comuns, também não é necessário inicializar um array em PHP, todavia, caso você o queira fazer, pode ser feito de duas formas “diretamente” ou utilizando a função **array()**:

```php
//inicialização 'direta'
$arr[]="";

//inicialização utilizando a função array()
$arr=array();
```

Populando array

- Novamente, os arrays podem ser populados “diretamente”, ex:

```
$arr[0]       = "valor";
$arr[1]       = "item";
$arr['chave'] = 115566;
```

Os índices podem ser numéricos ou string

- Ou utilizando a função **array()** e o operador **=>**, ex:

```
$arr = array(0 => "valor", 1 => "item", 'chave' => 115566);
```

- Ou ainda nas versões mais recentes:

```
$arr = [0 => "valor", 1 => "item", 'chave' => 115566];
```

Populando um array

- Como podemos observar, um mesmo array em PHP aceita elementos de diferentes tipos, inclusive, é possível ir criando novos arrays dentro do array, exemplo:

```php
$arr = array(
          0 => "valor",
          1 => "item",
          'chave' => 115566,
          'opcoes' => array(1,2,3)
       );
```

Acessando elementos do array

- Independente da forma como o array foi populado, seus elementos são acessados da mesma forma: *$nome_array[indice];*

```
echo "<br />item 0: ".$arr[0];
$soma = $arr['chave']*2;
echo "<br />n: ".strlen($arr[1]); //função que faz
a contagem de caracteres de uma string
```

Iterando um array

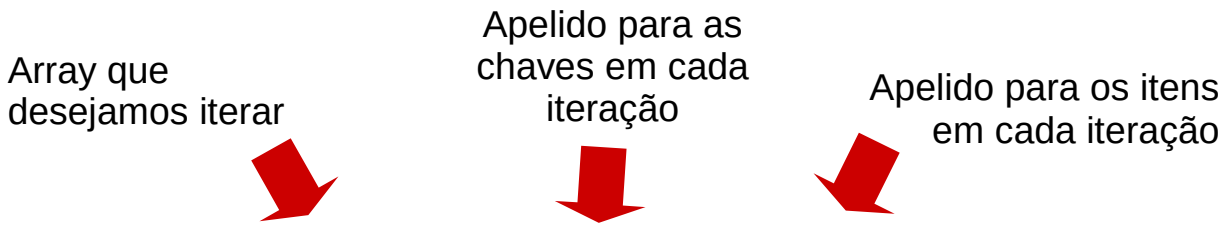
- Podemos iterar um array da forma “tradicional”:

```php
for ($i=0;$i<count($arr);$i++) {
   echo "<br /> "."i: ".$i." ".$arr[$i];
}
```

Essa forma de iteração **também funcionará** para chaves em string? **Não**, não funcionará!

Iterando um array

- No caso de array associativo (com chaves em string), o ideal é utilizar a função **`foreach`**:




```php
foreach ($arr as $k => $a) {
   echo "<br />chave: ".$k." valor: ".$a;
}
```

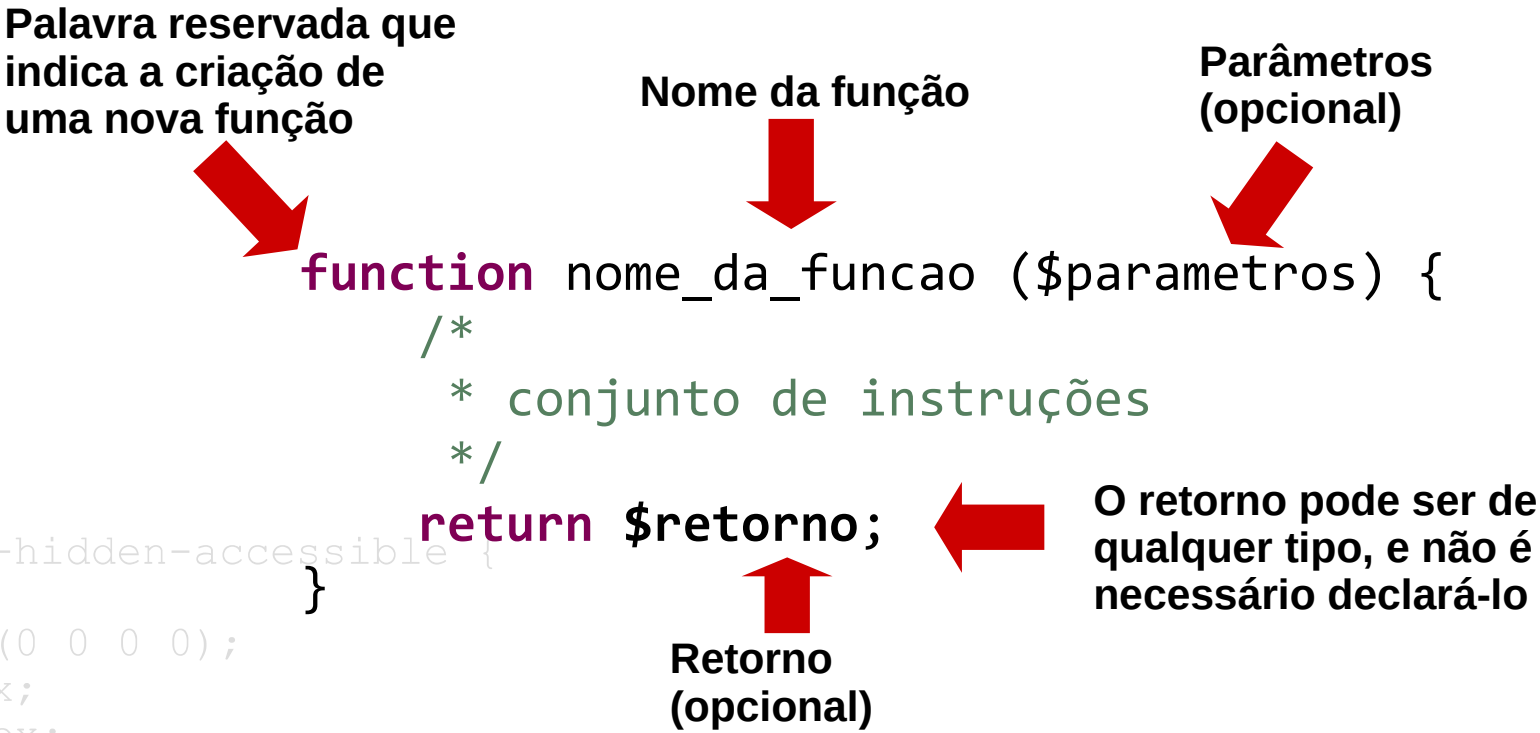
Ainda sobre arrays

- Como em muitas outras linguagens, listas e coleções em geral (arrays, vetores, etc) são um importante recurso do PHP e consequentemente, são amplamente utilizados pelos desenvolvedores. Há diversas funções em PHP para trabalharmos com array, é possível, por exemplo, trabalhar um array como um pilha, utilizando as funções *`array_push()`* e *`array_pop()`* ou ainda uma fila utilizando *`array_shift()`* e *`array_unshift()`*

Mais sobre arrays em PHP:
http://php.net/manual/pt_BR/ref.array.php
http://php.net/manual/pt_BR/language.types.array.php

Funções

- A sintaxe básica para para criação de funções em PHP é a seguinte:



Exemplo

```php
//função para imprimir n vezes
//uma palavra sempre aumentando a fonte
function imprime($str,$n) {
    $size=10;
    while ($n>) {
        echo "<p style='font-size: ".$size."px '>".$str."</p>";
        $n--;
        $size+=2;
    }
}

//exemplo de uso:
imprime("Jivago",30);
```

Exemplo, parâmetro opcional

```php
//soma simples de dois ou três parâmetros (terceiro é opcional)
function somaDoisTres($a,$b,$c=null) {
    $soma=$a+$b;
    if (isset($c)) {
        $soma+=$c;
    }
    return $soma;
}
```

Nesse caso, a função *somaDoisTres()* pode ser chamada com a passagem de dois ou três parâmetros. Se por passado apenas um parâmetro será gerado um ***warning.***

Variáveis Globais e SuperGlobais

- Assim como a maioria das linguagens de programação, PHP também possui escopo local e global de variáveis.
  - Toda variável inicializada dentro de uma função pertence ao escopo local da função
    - Mesmo assim, caso necessário é possível acessar as variáveis do escopo global do *script* utilizando a “Variável SuperGlobal” ***$GLOBALS***, ex: *`$GLOBALS['nome_da_variavel']`*

Exemplo

```php
<?php

   $professor="João";

   function exemplo() {
      echo "<br />escopo global: ".$GLOBALS['professor'];
      $GLOBALS['professor']="Jivago";
   }

   exemplo();

   echo "<br />Após chamada da função: ".$professor;

?>
```

Variáveis SuperGlobais

- Em **PHP**, Variáveis **SuperGlobais** são arrays (associativo) pré-definidos e disponíveis em qualquer ponto no escopo do *script*, as variáveis SuperGlobais existentes são:

$GLOBALS – Escopo global de variáveis
$_SERVER – Array com dados criados pelo servidor (ex: nome e local do script executado, porta, etc)
$_GET – Dados submetidos via GET
$_POST – Dados submetidos via POST
$_FILES – Arquivos (imagens, documentos, etc) submetidos via formulário
$_COOKIE – Parâmetros guardados em cookie
$_SESSION – Parâmetros guardados na sessão do servidor
$_REQUEST – "Atalho" para $_POST e $_GET
$_ENV – Informações sobre o ambiente de execução do interpretador PHP (ex: nome do usuário)

http://php.net/manual/pt_BR/language.variables.superglobals.php

Inclusão de Arquivo

- Tradicionalmente, o desenvolvimento de sistemas é dividido em vários arquivos (classes, formulários, controllers, etc), em PHP, costumeiramente essa divisão pode-se tornar maior ainda.
- Tornando corriqueira a inclusão de um arquivo com um trecho de código PHP dentro de outro arquivos php.
- Para isso, o PHP possui três funções básicas de inclusão de trecho de código de um arquivo em outro arquivo.

Inclusão de Arquivo

- Funções básicas para a inclusão de arquivo:
  - `include("nome_do_arquivo.php");`
  - `require("nome_do_arquivo.php");`
  - `require_once("nome_do_arquivo.php");`

A principal diferença entre o **include** e o **require** é que caso o arquivo não seja encontrado, na primeira função é emitido um ***warning*** e na segunda um ***fatal error.*** No caso da **require_once** ela ainda verifica se o arquivo já foi incluído alguma vez e em caso de resposta afirmativa, o arquivo não será incluído novamente.

Formulários

- Conforme vimos, formulários são importantes recursos do HTML/HTML 5 uma vez que fornecem aos usuários meios para a inserção (entrada) de dados em web sites e sistemas web.
- Outro importante aspecto dos formulários é que ao serem submetidos (dados enviados), os formulários geram requisições que são processadas do lado do servidor, fazendo uso dos recursos disponíveis.

Exemplo

```
<form name="form-aula" id="form-aula" method="POST" action="recebe-dados.php">
    <fieldset>
        <legend>Dados Pessoais</legend>
        <label for="txt-primeiro-nome">Primeiro nome:</label>
        <input id="txt-primeiro-nome" type="text" name="primeiro-nome" />
        <br />
        Segundo nome: <input id="segundo" type="text" name="segundo-nome" />
        <br />
        Sexo: M: <input type="radio" name="sexo" value="M" />
              F: <input type="radio" name="sexo" value="F" />
        <br />
    </fieldset>
    <fieldset>
        <legend>Endereço</legend>
        Endereço Completo:
        <br />
        <select name="lista-cidade">
            <option value="1">Cuiabá</option>
            <option value="2">Várzea Grande</option>
        </select>
        <br />
        <textarea cols="40" rows="4" name="endereco"></textarea>
    </fieldset>
    <input type="submit" value="Enviar" />
</form>
```

Explicando

1. `<form name="form-aula" id="form-aula" method="POST" action="recebe-dados.php">`

Arquivo que receberá a requisição com os dados do formulário

2. `<form name="form-aula" id="form-aula" method="POST" action="recebe-dados.php">`

Método de submissão dos dados do formulário (POST ou GET)

Nome do campo que estará na variável SuperGlobal $_POST ou $_GET (de acordo com o method)

3. `<input id="txt-primeiro-nome" type="text" name="primeiro-nome" />`

recebe-dados.php

```php
<?php

	if (isset($_POST['primeiro-nome']) && !empty($_POST['primeiro-nome'])) { //uma validação bem simples
		$nome_completo = $_POST['primeiro-nome']." ".$_POST['segundo-nome'];
	}
	else {
		$nome_completo = "Você não preencheu o primeiro nome!";
	}

	$sexo = $_POST['sexo'];
	if ($sexo=="F") {
		$sexo="Feminino";
	}
	else if ($sexo=="M") {
		$sexo="Masculino";
	}
	else {
		$sexo="-";
	}

	$cidade = $_POST["list-cidade"];
	if ($cidade==1) {
		$cidade="Cuiabá";
	}
	else if ($cidade==2) {
		$cidade="Várzea Grande";
	}
	else {
		$cidade="-";
	}

	$endereco = $_POST['endereco']."<br />".$cidade;

	echo "<h1>Seus Dados</h2>";
	echo "<p><strong>Nome completo:</strong> ".$nome_completo."</p>";
	echo "<p><strong>Sexo:</strong> ".$sexo."</p>";
	echo "<p><strong>Endereço:</strong><br />".$endereco."</p>";

?>
```

Resultado

Seus Dados

Nome completo: Jivago Medeiros

Sexo: Masculino

Endereço:
Av. Fernando Corrêa da Costa, Cuiabá
Cuiabá

Considerações

```
<select name="lista-cidade">
    <option value="1">Cuiabá</option>
        ...
```

```
$cidade = $_POST["list-cidade"];
```

Quando temos uma lista (***select***) em nosso formulário, o valor passado é o ***value*** do item selecionado

```
<input type="radio" name="sexo" value="M" />
<input type="radio" name="sexo" value="F" />
```

```
$sexo = $_POST['sexo'];
```

No caso de elemento HTML do tipo radio, o funcionamento é semelhante: séria submetido o ***value*** do item selecionado. Lembrando que quando tempos vários radios com o mesmo nome, só é possível selecionar um.

checkbox

Se estivéssemos utilizando ***checkbox*** em nosso formulário, que permitem a seleção de múltiplos itens, trabalharíamos com os dados que chegam no PHP de uma forma um pouco diferente, pois esses chegam em forma de **array**:

Exemplo no formulário HTML:

```
<input type="checkbox" name="opcoes" value="A" />
<input type="checkbox" name="opcoes" value="B" />
<input type="checkbox" name="opcoes" value="C" />
<input type="checkbox" name="opcoes" value="D" />
```

Exemplo no arquivo PHP:

```
<?php

    if (isset($_POST['opcoes']) && is_array($_POST['opcoes'])) {
        echo "<h3>Opções selecionadas:</h3>";
        foreach ($_POST['opcoes'] as $op) {
            echo "<p>".$op."</p>".
        }
    }
    else {
        echo "<br />Você não selecionou nenhuma opção!"
    }
?>
```

Outras Considerações

- Os dados de um formulário podem ser submetidos para a mesma página, basta deixar o atributo **action** em branco, colocar o próprio nome do arquivo que está o formulário, ou ainda utilizar a propriedade **PHP_SELF** da variável super global **$_SERVER**

Exemplo:

```
<form name="form-aula" id="form-aula"
        method="POST" action="<?php echo $_SERVER['PHP_SELF'] ?>">
```

(lembrando que o arquivo com o formulário terá que estar com a extensão .php)

Outras Considerações

- É possível submeter via formulário dados por POST e por GET ao mesmo tempo

Exemplo:

Os dados dos campos do formulário são submetidos via **POST**

```
<form name="form-aula" id="form-aula"  method="POST"
                              action="arquivo.php?acao=cadastrar">
```

E também é enviada uma variável pela URL (via GET), ***$_GET['acao']***