

PHP - Hypertext Preprocessor

O HTML é limitado

- ▶ Apesar de toda evolução do HTML e dos navegadores atuais, a Web ainda é um ambiente bastante restrito
- ▶ Todo projeto Web sério deve ir além do HTML, CSS e JavaScript
- ▶ Existem diversas linguagens que podem ser usadas no ambiente Web



Ruby



python

Microsoft
C#.net

- ▶ Com o uso de uma linguagem de programação, é possível:
 - ▶ gerar páginas dinamicamente com dados de um banco de dados
 - ▶ enviar *emails* para usuários
 - ▶ processar tarefas complexas
 - ▶ garantir validações de segurança
 - ▶ e muito mais!



INSTITUTO FEDERAL
GOIANO

PHP - Hypertext Preprocessor
Prof. Marcos Alves Vieira
marcos.vieira@ifgoiano.edu.br

Páginas estáticas vs. páginas dinâmicas

- ▶ Página estática
 - ▶ Todo conteúdo é fixo
 - ▶ Não existe interação entre o usuário e a página
- ▶ Página dinâmica
 - ▶ O conteúdo é gerado a cada visita (requisição) à página
 - ▶ Bancos de dados, acesso a fontes externas, etc...
 - ▶ Possibilita interação entre o usuário e a página
 - ▶ Formulários

O que o PHP pode fazer?

- ▶ Scripts do lado do servidor (*server-side*)
 - ▶ Interpretador PHP (CGI)
 - ▶ Servidor Web
 - ▶ Navegador Web
- ▶ Scripts em linha de comando
 - ▶ Não necessita de navegador
- ▶ Aplicações desktop
 - ▶ PHP-GTK

O que o PHP pode fazer? *(cont.)*

- ▶ Geração dinâmica de diversos tipos de arquivos, além de HTML
 - ▶ Imagens, PDF, Animações Flash, XML, ...
- ▶ Conexão com banco de dados
- ▶ Comunicação com outros serviços
 - ▶ LDAP, IMAP, SMTP, POP3, HTTP, COM (Windows)
- ▶ E muito mais!

Características do PHP

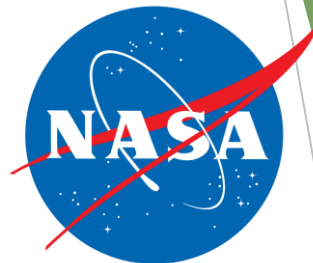
- ▶ Software livre
- ▶ Portabilidade
 - ▶ Multiplataforma
- ▶ Simplicidade
 - ▶ Rápido desenvolvimento
- ▶ Adaptabilidade
 - ▶ Procedural e/ou Orientado a Objetos
- ▶ Performance

O site oficial do PHP é <http://www.php.net>

Lá você encontra *downloads*, código fonte do PHP e um manual completo da linguagem, incluindo comentários de usuários.

Quem utiliza PHP?

facebook



Baidu 百度

YAHOO!

Google



ebay

SONY



flickr

sourceforge

W3C

O que é preciso para o PHP funcionar?

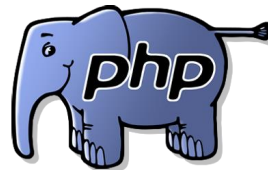
- ▶ Servidor web
 - ▶ Apache HTTP Server
 - ▶ Microsoft IIS
 - ▶ Zend Server



zend[®]Server



- ▶ Interpretador PHP

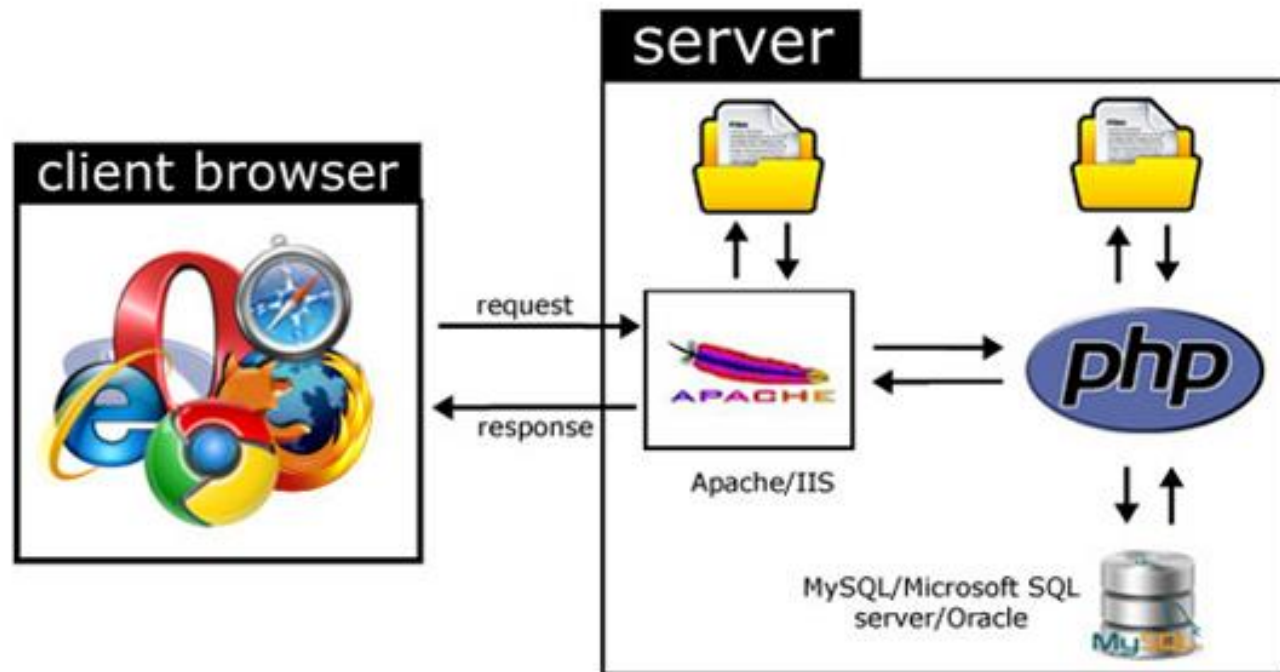


- ▶ Navegador Web



Como o PHP funciona?

Basic Structure



= File System



= Database

webdevtutes.blogspot.com



INSTITUTO FEDERAL
GOIANO

PHP - Hypertext Preprocessor
Prof. Marcos Alves Vieira
marcos.vieira@ifgoiano.edu.br

O que é um servidor web?

- ▶ Programa responsável por aceitar requisições HTTP, trata-las e, posteriormente, devolver a resposta para o requisitante
 - ▶ As requisições são geralmente originadas de navegadores (*browsers*) web
 - ▶ A resposta é devolvida em formato HTML e pode incluir objetos embutidos
 - ▶ Imagens
 - ▶ Vídeos
 - ▶ Áudio
 - ▶ Demais tipos de arquivo (PDF, XML, etc...)
- ▶ Exemplos de servidores web:
 - ▶ Apache, Tomcat, JBoss, IIS, nginx e outros

NGINX



APACHE

Facilite sua vida: simplifique

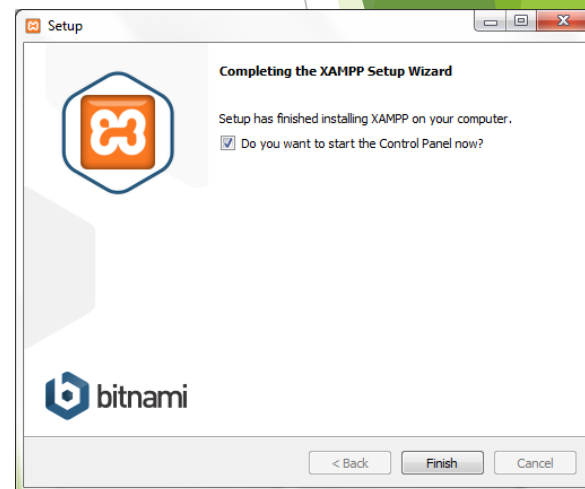
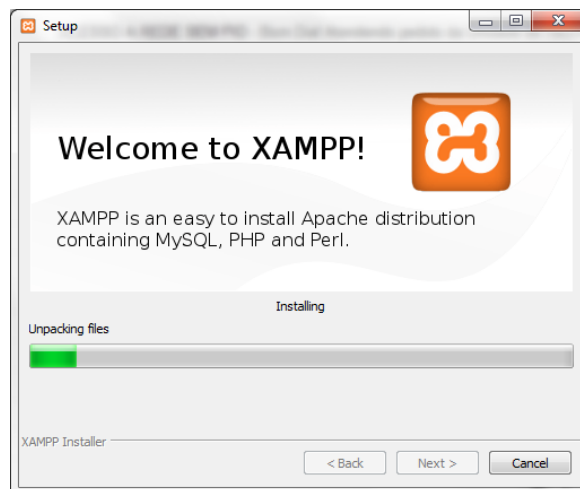
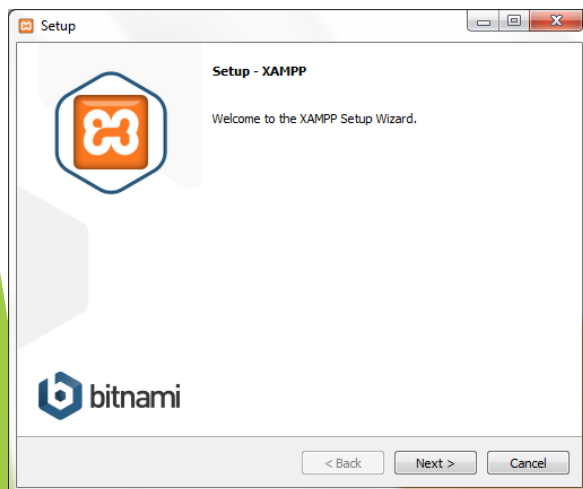
- ▶ XAMPP: Apache + MySQL + PHP + Perl
 - ▶ www.apachefriends.org
 - ▶ Multiplataforma (Windows, Linux e OS X)
 - ▶ **MUITO** fácil de instalar
 - ▶ Inclui phpMyAdmin



XAMPP

Instalação

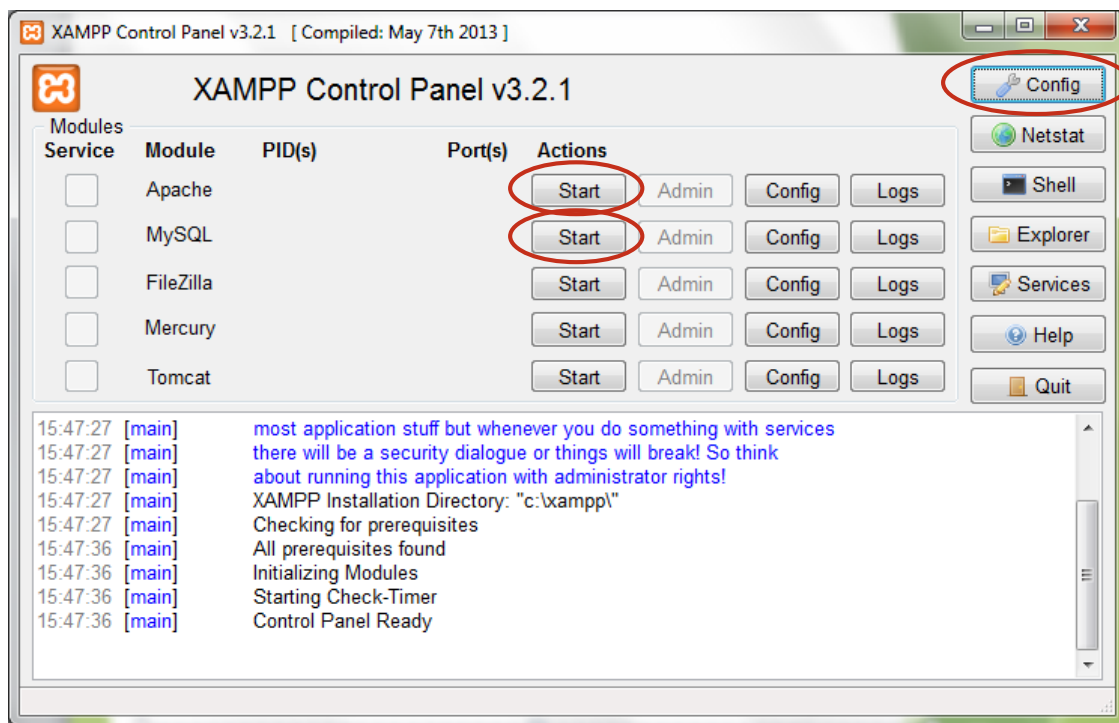
- ▶ Muita atenção na instalação supercomplicada:
 - ▶ Next, Next ... Finish



XAMPP

Configuração

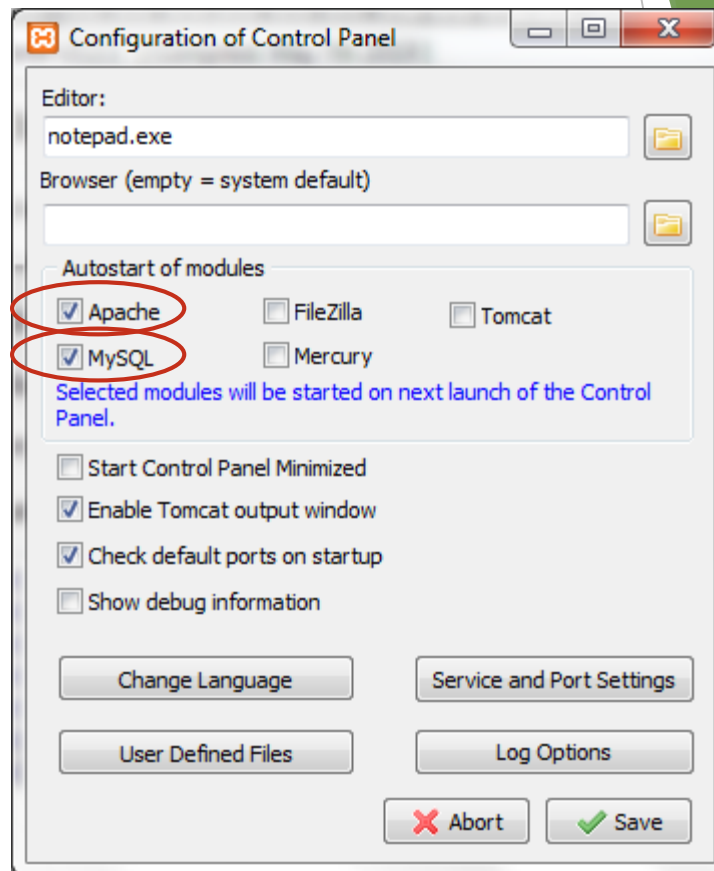
- Iniciar os módulos Apache e MySQL e clicar em "Config"



XAMPP

Configuração (continuação)

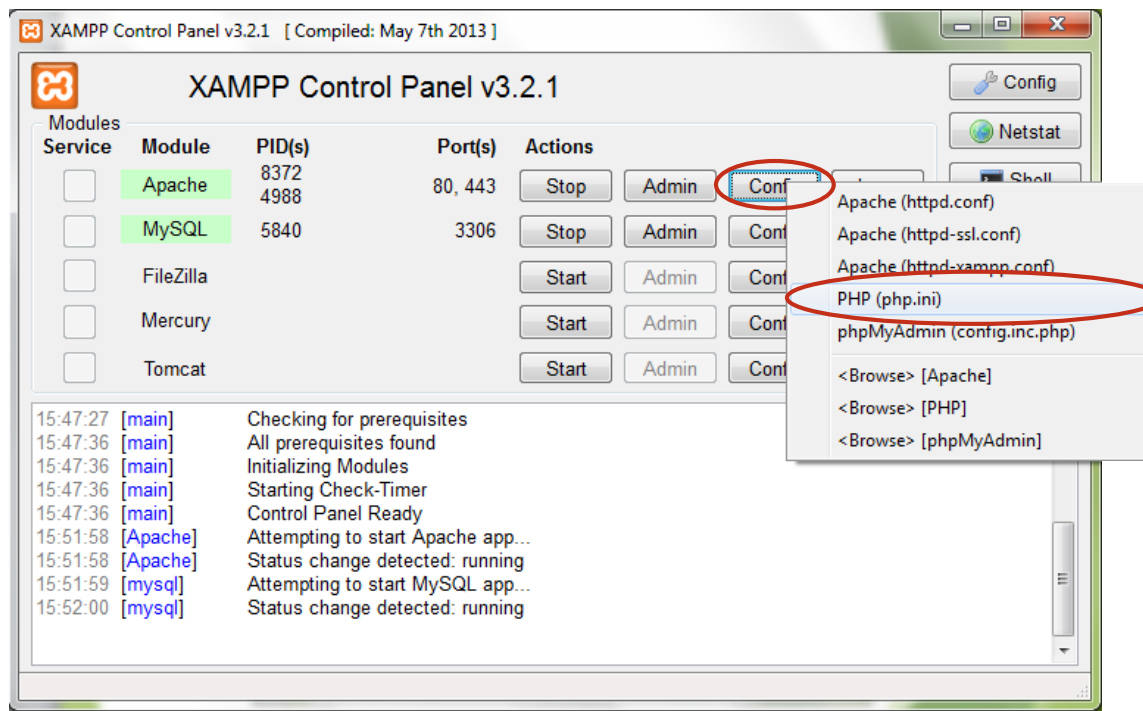
- ▶ Marcar os módulos para início automático:
 - ▶ Apache e MySQL
- ▶ Clicar em "Save"



XAMPP

Configuração (continuação)

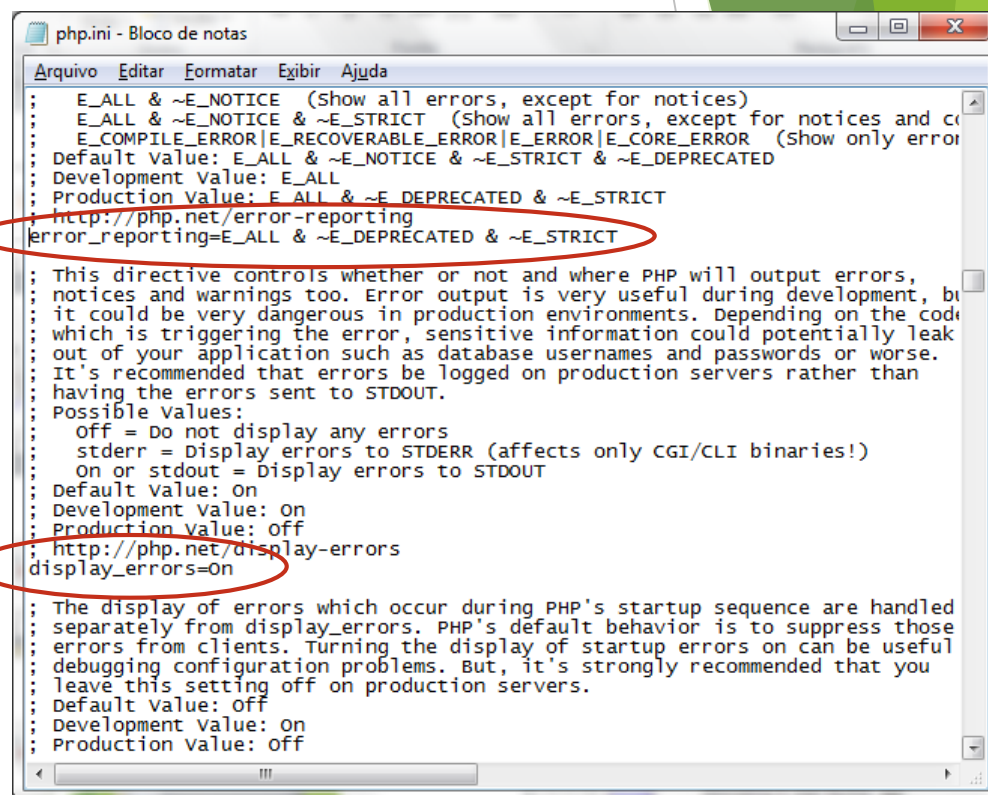
► Editar o arquivo php.ini



XAMPP

Configuração (continuação)

- ▶ Confirmar se as diretivas **error_reporting** e **display_errors** estão ativas (*descomentadas*)
- ▶ Caso seja necessário editar algo
 - ▶ Salvar o arquivo e reiniciar o Apache
 - ▶ Pelo console do XAMPP
 - ▶ Botão "Stop", depois "Start"



```
php.ini - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda
; E_ALL & ~E_NOTICE (Show all errors, except for notices)
; E_ALL & ~E_NOTICE & ~E_STRICT (Show all errors, except for notices and c
; E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR (Show only error
; Default value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED
; Development value: E_ALL
; Production value: E_ALL & ~E_DEPRECATED & ~E_STRICT
; http://php.net/error-reporting
error_reporting=E_ALL & ~E_DEPRECATED & ~E_STRICT

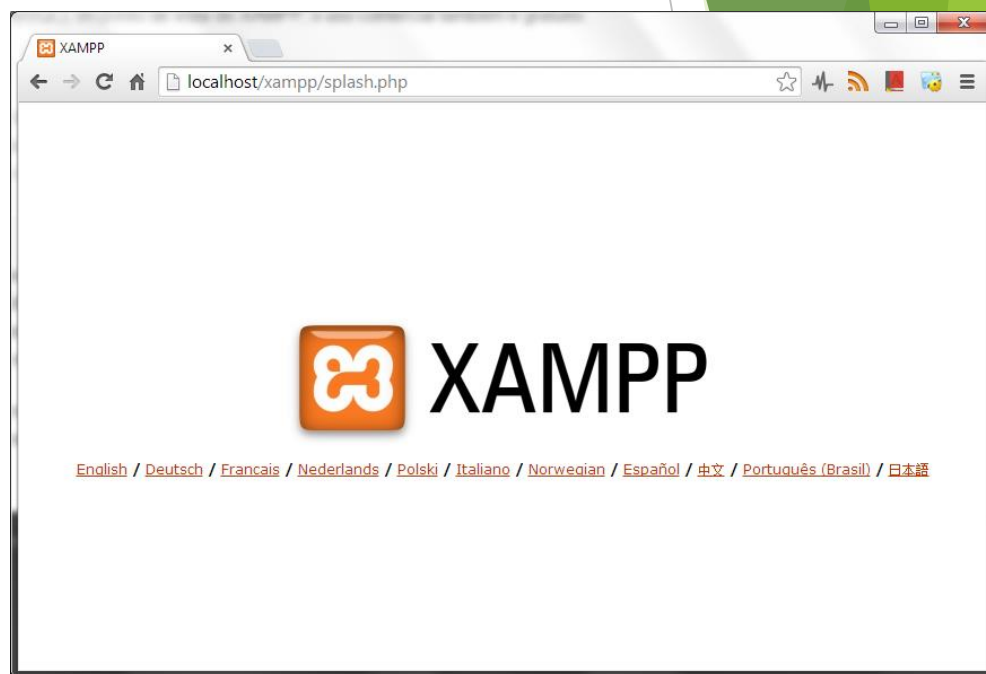
; This directive controls whether or not and where PHP will output errors,
; notices and warnings too. Error output is very useful during development, b
; it could be very dangerous in production environments. Depending on the cod
; which is triggering the error, sensitive information could potentially leak
; out of your application such as database usernames and passwords or worse.
; It's recommended that errors be logged on production servers rather than
; having the errors sent to STDOUT.
; Possible values:
;   off = Do not display any errors
;   stderr = Display errors to STDERR (affects only CGI/CLI binaries!)
;   On or stdout = Display errors to STDOUT
; Default value: On
; Development value: On
; Production value: off
; http://php.net/display-errors
display_errors=on

; The display of errors which occur during PHP's startup sequence are handled
; separately from display_errors. PHP's default behavior is to suppress those
; errors from clients. Turning the display of startup errors on can be useful
; debugging configuration problems. But, it's strongly recommended that you
; leave this setting off on production servers.
; Default value: off
; Development value: on
; Production value: off
```


XAMPP

Teste

- ▶ Abra o navegador e acesse a URL:
 - ▶ <http://localhost>
- ▶ Pronto! Tudo está OK :-)
- ▶ Os arquivos (PHP, HTML, imagens...) devem ser colocados na pasta
 - ▶ C:\xampp\htdocs
- ▶ Caso queira, pode apagar todo conteúdo inicial desta pasta



Como editar arquivos PHP?

► Editores de texto

- Bloco de notas
- Notepad++
- gedit
- vim



► IDE (*Integrated Development Environment*)

- Eclipse
- NetBeans
- ZendStudio (\$\$\$\$\$\$\$\$\$\$)
- Aptana



NetBeans

zendstudio



aptana

Delimitadores de código

- ▶ O código de um programa PHP deve estar contido entre os delimitadores:

```
<?php  
    //código;  
    //código;  
    //código;  
?>
```

Obs.: os comandos devem ser delimitados com ponto-e-vírgula (;)

Exercício 1

- a) Crie o arquivo `ex1.php` com o seguinte conteúdo:

```
<?php
    echo "Olá, mundo!";
?>
```

- b) Salve na pasta raiz de documentos do seu servidor web (`C:\xampp\htdocs`)

- c) Abra o navegador e acesse o endereço:

```
http://localhost/ex1.php
```

- d) Visualize o código-fonte da página gerada (Ctrl+U). Onde está o código PHP?

- e) Altere o código do arquivo, adicionando *tags* HTML:

```
<?php
    echo "<p><b>Olá</b>, <u>mundo</u>!</p>";
?>
```

- f) Acesse novamente o arquivo pelo navegador e visualize o código-fonte

Comentários

► Comentários de uma linha

- Utilizamos duas barras (//) ou o símbolo #

```
<?php
    //Este código imprime o texto 'Olá, mundo'
    echo "Olá, mundo!";
?>
```

► Comentários de múltiplas linhas

- Devem estar entre /* e */

```
<?php
    /*Este código imprime
    o texto 'Olá, mundo'*/
    echo "Olá, mundo!";
?>
```

Nota

Comentários são ignorados pelo interpretador do PHP

Comandos de saída (*output*)

- ▶ Estes comandos são utilizados para gerar uma saída na tela
 - ▶ Se o PHP for utilizado na linha de comando (*prompt* do sistema), a saída será no próprio console
 - ▶ Se o programa for executado via servidor web, a saída será exibida na página HTML gerada

- **echo**

Imprime uma ou mais variáveis ou *strings*

```
echo "Olá, mundo!"
```

- **print**

Imprime uma variável ou *string*

```
print "Olá, mundo!"
```

Exercício 2

Podemos mesclar trechos de código PHP com HTML

a) Crie o arquivo `ex2.php` com o seguinte conteúdo e salve na pasta raiz de documentos do servidor web:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Aprendendo PHP</title>
5 <meta charset="utf-8">
6 </head>
7 <body>
8 <h1>Mesclando HTML com PHP</h1>
9 <?php
10     echo "<p>Primeira linha</p>";
11     echo "<p>Segunda linha</p>";
12     echo "<p>Terceira linha</p>";
13 ?>
14 </body>
15 </html>
```

b) Acesse o arquivo pelo navegador e visualize o código-fonte



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Aprendendo PHP</title>
5 <meta charset="utf-8">
6 </head>
7 <body>
8 <h1>Mesclando HTML com PHP</h1>
9 <p>Primeira linha</p><p>Segunda linha</p><p>Terceira linha</p></body>
10 </html>
```

Variáveis

- ▶ São identificadores para representar valores mutáveis e voláteis
- ▶ Só existem durante a execução do programa
 - ▶ Armazenadas na memória RAM
- ▶ Para criar uma variável em PHP, devemos atribuir um nome, precedido do caractere cifrão (\$)

```
<?php
    $nome = "João";
    $sobrenome = " da Silva";
    echo $nome . $sobrenome;
?>
```

Concatenação: Unir o conteúdo de duas *strings*.

Em PHP, o operador de concatenação é o ponto (.)

Observação

PHP é uma linguagem **fracamente tipada**. Isto significa que não é necessário expressar o tipo das variáveis.

Variáveis *(continuação)*

Algumas dicas para nomenclatura de variáveis:

- ▶ Nunca inicie com números
- ▶ Nunca utilize espaços em brancos no meio
- ▶ Nunca utilize caracteres especiais (!@#\$%&* () [] {})
- ▶ Nomes de variáveis devem ser significativos e transmitir a ideia de seu conteúdo
- ▶ Utilizar preferencialmente letras em minúsculo
 - ▶ Em caso de mais de uma palavra, separe pelo caractere *underline* (_) ou use as iniciais em maiúsculo

```
<?php
    $codigo_cliente = 12345;
    $codigoCliente = 12345;
?>
```

Observação

O PHP é *case sensitive*, isto é,
\$codigo é diferente de \$Codigo

Exercícios 4 e 5

- 4) Teste a criação e concatenação de variáveis. Salve como `ex4.php`

Exemplo 1:

```
<?php
$primeiro_nome = "Marcos";
$nome_do_meio = "Alves";
$sobrenome = "Vieira";
echo $primeiro_nome . $nome_do_meio . $sobrenome;
?>
```

Exemplo 2:

```
<?php
$primeiro_nome = "Marcos";
$nome_do_meio = "Alves";
echo $primeiro_nome . " " . $nome_do_meio;
?>
```

Use sua criatividade para fazer mais testes.

Aproveite para ver o que acontece quando você esquece um ponto-e-vírgula ou o cifrão.

- 5) Crie o arquivo `ex5.php` e teste o uso de variáveis, mesclando código PHP com HTML. Experimente fazer diversos trechos PHP (`<?php ... ?>`) e responda a si mesmo a pergunta: É possível utilizar uma variável criada em outro trecho PHP, dentro do mesmo arquivo?

Dica: Utilize como base o código do arquivo `ex3.php`

Tipos de dados

Booleano

- ▶ Expressa um valor lógico que pode ser *verdadeiro* ou *falso*
- ▶ Para especificar um valor booleano, deve-se utilizar as palavras TRUE ou FALSE

```
<?php
    $estou_feliz = TRUE;
    $vai_chover = ($umidade > 90);
?>
```

Repare que o valor TRUE (ou FALSE) do tipo booleano não deve estar entre parênteses.

- ▶ Também são considerados valores *falsos* em comparações booleanas:
 - ▶ Inteiro 0
 - ▶ Ponto flutuante 0.0
 - ▶ Uma *string* vazia "" ou "0"
 - ▶ Um *array* vazio
 - ▶ Um objeto sem elementos
 - ▶ Tipo NULL
- ▶ Qualquer valor diferente dos acima é considerado como *verdadeiro*

Tipos de dados

Numérico

- Tipos numéricos podem ser especificados em notação decimal (base 10), hexadecimal (base 16) ou octal (base 8), precedido ou não de sinal (+ ou -)

```
<?php
//número decimal
$a = 1234;
//número negativo
$a = -1234;
//número octal (equivalente a 83 em decimal)
$a = 0123;
//número hexadecimal (equivalente a 26 em decimal)
$a = 0x1A;
//ponto flutuante
$a = 1.234;
//notação científica
$a = 4e23;
?>
```

Tipos de dados

String

- ▶ Uma *string* é uma cadeia de caracteres alfanuméricos
- ▶ Para declarar uma *string*, podemos usar aspas simples (' ') ou duplas (" ")

```
<?php
    $variavel = 'Isto é um teste';
    $variavel = "Isto é um teste";
?>
```

Tipos de dados

Array ou vetor

- ▶ *Array* é uma lista de valores
- ▶ Cada elemento de um *array* pode ser de um tipo diferente (números, *strings*, objetos)
- ▶ Um *array* pode crescer dinamicamente

```
<?php
    $carros = array("Palio", "Corsa", "Gol");
    echo $carros[1];      //resultado = "Corsa"
?>
```

Nota

Arrays serão tratados com maiores detalhes futuramente.

Tipos de dados

NULL

- ▶ A utilização do valor especial NULL significa que a variável não tem valor
- ▶ NULL é o único valor possível do tipo NULL

```
<?php
    $valor1 = NULL;    //não há valor algum
    $valor2 = 100;     //valor decimal inteiro 100
    $valor3 = 0;       //valor decimal inteiro 0
?>
```

Tipos de dados

Constante

- ▶ Valor que não sofre modificações durante a execução do programa
- ▶ Só pode conter valores escalares
 - ▶ Booleano, inteiro, ponto flutuante e *string*
 - ▶ Não pode conter outros valores, como vetores e objetos
- ▶ Seguem as mesmas regras de nomenclatura das variáveis
 - ▶ Com exceção de que não devem ser precedidas de \$
 - ▶ Geralmente utiliza-se nomes em maiúsculo

```
<?php
define("NUMERO_PI", 3.14159265359);
$raio = 10;
echo "A circunferência é " . 2 * NUMERO_PI * $raio;
?>
```


Operadores

Atribuição

- ▶ Utilizado para atribuir valor a uma variável
- ▶ O operador básico de atribuição é o sinal de igual (=)

```
<?php
    $var = 10; //atribui o valor 10 a $var
    $var += 5; //soma 5 em $var
    $var -= 5; //subtrai 5 em $var
    $var *= 5; //multiplica $var por 5
    $var /= 5; //divide $var por 5
    $var %= 3; //$var recebe o resto de sua divisão por 3
?>
```

Operadores

Atribuição (*continuação*)

► Pré e pós incremento/decremento:

Operadores	Descrição
++\$a	Pré-incremento: incrementa \$a em 1 e, então, retorna \$a
\$a++	Pós-incremento: retorna \$a e, então, incrementa \$a em 1
--\$a	Pré-decremento: decrementa \$a em 1 e, então, retorna \$a
\$a--	Pós-decremento: retorna \$a e, então, decrementa \$a em 1

```
<?php
    $teste1 = 10;      //atribui o valor 10 a $teste1
    $teste2 = 10;      //atribui o valor 10 a $teste2

    echo --$teste1;    //imprime 9 e $teste1 vale 9
    echo $teste2--;    //imprime 10 e $teste2 vale 9
?>
```

Exercício 7

Teste livremente TODOS os operadores de atribuição apresentados, incluindo pré e pós incremento/decremento.

Salve seu arquivo como `ex7.php`

Exemplo 1:

```
<?php
    $nome = "Felipe";
    $idade = 17;
    echo "<p>Eu sou $nome e tenho $idade anos</p>";
?>
```

Exemplo 2:

```
<?php
    $var = 100;
    $var += 25; //soma 25 em $var
    echo $var++;
    echo $var;
?>
```

Dica

Use diferentes tipos de dados, como números inteiros, ponto flutuante, constantes, *strings*...

Operadores Aritméticos

- Utilizados para realização de cálculos matemáticos

Sintaxe	Significado
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto da divisão)

- Pode-se também utilizar parênteses para alterar as prioridades aritméticas

```
<?php
    $a = 2;
    $b = 4;
    echo $a+3*4+5*$b;           //resultado 34
    echo ($a+3)*4+(5*$b);       //resultado 40
?>
```

Operadores

Aritméticos *(continuação)*

- O PHP realiza automaticamente a conversão de tipos em operações:

```
<?php
//declaração de string contendo 10
$a = '10';

//soma a variável $a com 5
$soma = $a + 5;

//exibe o resultado: 15
echo $soma;
?>
```

Para esta operação, o valor da variável `$a` é convertido para inteiro.

Exercício 8

Escreva um programa com duas variáveis numéricas e imprima:

- a) sua soma
- b) sua multiplicação
- c) sua divisão
- d) o resto de sua divisão
- e) a média entre eles

Salve seu arquivo como `ex8.php`

Operadores Relacionais

- ▶ Utilizados para realizar comparações entre valores ou expressões
- ▶ O resultado é sempre um valor booleano: `TRUE` ou `FALSE`

Operadores	Descrição
<code>==</code>	Igual: resulta verdadeiro (<code>TRUE</code>) se expressões forem iguais
<code>===</code>	Igual e do mesmo tipo de dados
<code>!=</code> ou <code><></code>	Diferente: resulta verdadeiro (<code>TRUE</code>) se as expressões forem diferentes
<code><</code>	Menor que
<code>></code>	Maior que
<code><=</code>	Menor ou igual que
<code>>=</code>	Maior ou igual que

Operadores

Relacionais - Exemplos

- Uso errado do operador de atribuição para realizar uma comparação:

```
<?php
    if($a = 5){
        echo "essa operação atribui 5 à variável a e retorna verdadeiro";
    }
?>
```

- Comparação de duas variáveis (inteiro e *string*):

```
<?php
    $a = 1234;
    $b = '1234';

    if($a == $b) {
        echo '$a e $b são iguais';
    }

    if($a != $b) {
        echo '$a e $b são diferentes';
    }
?>
```


Operadores

Relacionais - Exemplos (*continuação*)

- Declaração de duas variáveis (inteiro e *string*) e de seus tipos:

```
<?php
    $c = 1234;
    $d = '1234';

    if($c === $d) {
        echo '$c e $d são iguais e do mesmo tipo';
    }

    if($c !== $d) {
        echo '$c e $d são de tipos diferentes';
    }
?>
```

Operadores

Relacionais - Exemplos *(continuação)*

- O PHP considera o valor zero como falso em comparações lógicas

```
<?php
    $e = 0;

    //zero sempre é igual a FALSE
    if($e == FALSE) {
        echo '$e é FALSO';
    }

    //testa se $e é do tipo FALSE
    if($e === FALSE) {
        echo '$e é do tipo FALSE';
    }

    //testa se $e é igual e do mesmo tipo de zero
    if($e === 0) {
        echo '$e é zero mesmo';
    }
?>
```

Operadores Lógicos

- ▶ Utilizados para combinar expressões lógicas entre si
- ▶ O resultado é sempre um valor booleano: `TRUE` ou `FALSE`

Operadores	Descrição
<code>(\$a and \$b)</code> ou <code>(\$a && \$b)</code>	E: verdadeiro (<code>TRUE</code>) se tanto <code>\$a</code> quanto <code>\$b</code> forem verdadeiros
<code>(\$a or \$b)</code> ou <code>(\$a \$b)</code>	OU: verdadeiro (<code>TRUE</code>) se <code>\$a</code> ou <code>\$b</code> forem verdadeiros
<code>(\$a xor \$b)</code>	XOR: verdadeiro (<code>TRUE</code>) se <code>\$a</code> ou <code>\$b</code> forem verdadeiros, de forma exclusiva
<code>(! \$a)</code>	NOT: verdadeiro (<code>TRUE</code>) se <code>\$a</code> for <code>FALSE</code>

Operadores Lógicos - Exemplos

- Se `$vai_chover` e `$esta_frio` forem ambas verdadeiras, imprime "Não vou sair de casa"

```
<?php
    $vai_chover = TRUE;
    $esta_frio = TRUE;

    if($vai_chover and $esta_frio){
        echo "Não vou sair de casa";
    }
?>
```

- Se uma variável for `TRUE` e a outra `FALSE`, imprime "Vou pensar duas vezes antes de sair de casa"

```
<?php
    $vai_chover = FALSE;
    $esta_frio = TRUE;

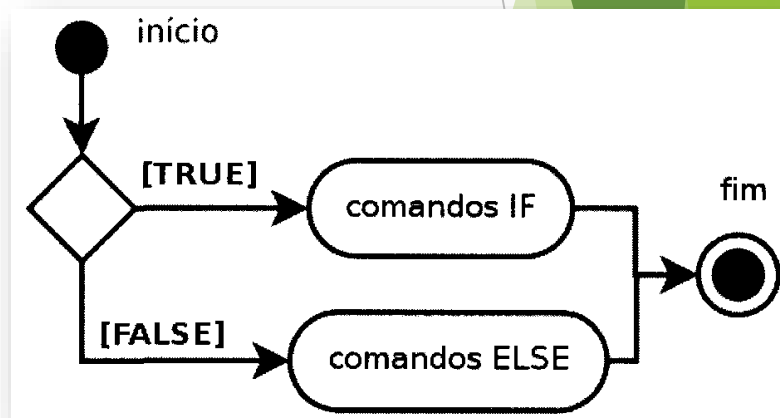
    if($vai_chover xor $esta_frio){
        echo "Vou pensar duas vezes antes de sair de casa";
    }
?>
```

Estruturas de controle

IF

- ▶ Realiza um desvio condicional na execução natural do programa
- ▶ Caso a condição seja satisfeita, serão executadas as instruções do bloco de comandos IF
- ▶ Caso a condição não seja satisfeita, o bloco é ignorado e as instruções do bloco ELSE são executadas (se houver)

```
if(expressão) {  
    comandos se expressão é verdadeira  
}else{  
    comandos se expressão é falsa  
}
```



Estruturas de controle

IF - Exemplos

- ▶ Quando não especificamos o operador lógico, o PHP retorna TRUE sempre que a variável tiver um conteúdo válido:

```
<?php
    $a = 'conteúdo';

    if($a){
        echo '$a tem conteúdo';
    }else{
        echo '$a não tem conteúdo';
    }

    if($b){
        echo '$b tem conteúdo';
    }
?>
```

Estruturas de controle

IF - Exemplos *(continuação)*

- Podemos utilizar um IF aninhados ou usar operadores lógicos:

```
<?php
    $salario = 1020;
    $tempo_servico = 12;
    $tem_reclamacoes = FALSE;

    if($salario>1000){
        if($tempo_servico >= 12){
            if($tem_reclamacoes != TRUE){
                echo 'Parabéns, você foi promovido!';
            }
        }
    }

    if(($salario > 1000) and ($tempo_servico >= 12) and
($tem_reclamacoes != TRUE)){
        echo 'Parabéns, você foi promovido!';
    }
?>
```

Estruturas de controle

IF - Exemplos *(continuação)*

- Atribuição condicional à uma variável

```
if($valor_venda > 100){  
    $resultado = 'muito caro';  
}else{  
    $resultado = 'pode comprar';  
}
```

- O mesmo código acima pode ser escrito utilizando o conceito de **Operador Ternário**

```
$resultado = ($valor_venda > 100) ? 'muito caro' : 'pode comprar';
```

- A sintaxe do **Operador Ternário** é a seguinte:

```
(<condição>) ? <instruções para verdadeiro> : <instruções para falso>
```


Exercícios 9, 10 e 11

- 9) Criar uma variável com valor numérico e informar se é divisível por 10, por 5, por 2 ou se não é divisível por nenhum destes

Salve seu arquivo como `ex9.php`

- 10) Criar duas variáveis numéricas e imprimi-las em ordem decrescente (suponha números diferentes)

Salve seu arquivo como `ex10.php`

- 11) Criar uma variável numérica entre 1 e 12 e escrever o mês correspondente. Caso o número seja fora desse intervalo, informar que não existe mês com este número

Salve seu arquivo como `ex11.php`

Estruturas de controle

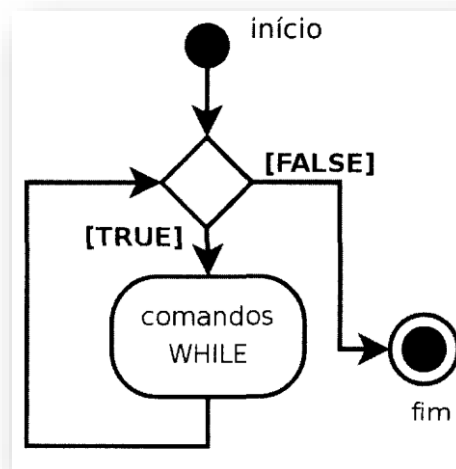
WHILE

- ▶ Estabelece um laço de repetição
- ▶ O bloco de comando será executado repetidamente enquanto a condição for verdadeira

```
while (expressão) {  
    comandos  
}
```

Exemplo

```
<?php  
    $a = 1;  
  
    while ($a < 5) {  
        echo $a;  
        $a++;  
    }  
?>
```



Estruturas de controle

FOR

- Laço de repetição baseado em três parâmetros

```
for(expr1; expr2; expr3) {  
    comandos  
}
```

Parâmetros	Descrição
expr1	Executado antes do início do laço de repetição. Geralmente possui uma variável contadora
expr2	Condição de execução: enquanto verdadeira (TRUE), o bloco de comandos será executado
expr3	Executado a cada execução. Geralmente utilizado para incrementar a variável contadora

- Exemplo

```
<?php  
    for($i = 1; $i < 5; $i++) {  
        echo $i;  
    }  
?>
```

Exercícios 12, 13 e 14

12) Criar uma variável numérica e imprimir a palavra "IFGoiano" a quantidade de vezes informada nesta variável.

Salve seu arquivo como `ex12.php`

13) Criar duas variáveis com valor inteiro e imprimir todos os números pares contidos em seu intervalo.

Dica: considere que a primeira variável é sempre menor que a segunda

Salve seu arquivo como `ex13.php`

14) Criar uma variável com valor inteiro positivo e imprimir todos os números de 1 até ela. Ao final, imprimir a soma dos números do intervalo.

Salve seu arquivo como `ex14.php`

Estruturas de controle

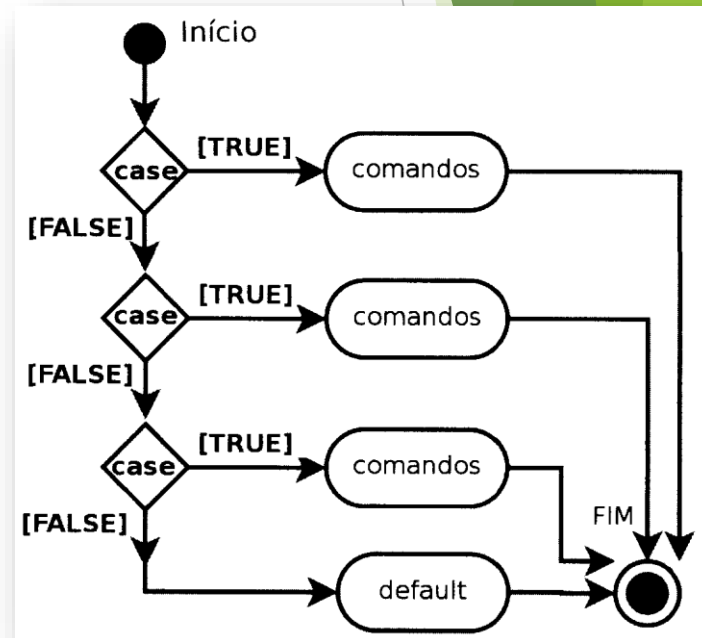
SWITCH

- ▶ Simula uma bateria de testes sobre uma variável
- ▶ Similar a uma série de comandos IF sobre a mesma expressão

```
switch($variável){  
  case "valor 1":  
    //comandos  
    break;  
  case "valor 2":  
    //comandos  
    break;  
  case "valor 3":  
    //comandos  
    break;  
  default:  
    //comandos  
}
```

Obs1: O bloco default é opcional e é executado caso a expressão não se encaixe em nenhum dos "casos".

Obs2: O comando break aborta a execução do bloco de comandos.



Estruturas de controle

SWITCH - Exemplos

- Estes exemplos apresentam duas formas de atingir o mesmo resultado usando IF e usando a estrutura SWITCH

```
<?php
$i = 1;
if($i == 0) {
    echo 'i é igual a 0';
} else if($i == 1) {
    echo 'i é igual a 1';
} else if($i == 2) {
    echo 'i é igual a 2';
} else {
    echo 'i não é 0, 1 ou 2';
}
?>
```

```
<?php
$i = 1;
switch($i){
    case 0:
        echo 'i é igual a 0';
        break;
    case 1:
        echo 'i é igual a 1';
        break;
    case 2:
        echo 'i é igual a 2';
        break;
    default:
        echo 'i não é 0, 1 ou 2';
}
?>
```

Exercício 15

Criar uma variável numérica com valor entre 1 e 12 e escrever o mês correspondente. Caso o número seja fora desse intervalo, informar que não existe mês com este número

Salve seu arquivo como `ex15.php`

Observação: este exercício é idêntico ao exercício 11, só que agora você deve utilizar SWITCH para resolvê-lo.

Estruturas de controle

FOREACH

- ▶ Laço de repetição para iterações em *arrays*
- ▶ É um FOR simplificado que decompõe o *array* em cada um de seus elementos usando a cláusula *AS*

```
foreach($array as $valor){  
    instruções  
}
```

▶ Exemplo

```
<?php  
    $fruta = array('maçã', 'laranja', 'pera', 'banana');  
  
    foreach($fruta as $valor){  
        echo "$valor - ";  
    }  
?>
```


Exercícios 16 e 17

16) Criar um vetor (*array*) com 10 números inteiros e imprimir uma mensagem contendo o valor de cada número armazenado no vetor e informando se este é par ou ímpar.

Salve seu arquivo como `ex16.php`

17) Criar 2 vetores com 10 números inteiros cada. Imprimir os números que estão no primeiro vetor e NÃO estão no segundo vetor.

Observação: não repetir os números na hora de imprimir!

Salve seu arquivo como `ex17.php`

Funções

- ▶ Uma função é um pedaço de código com objetivo específico
- ▶ Pode receber um conjunto de parâmetros e retornar um dado
- ▶ Uma vez declarada, pode ser utilizada inúmeras vezes
- ▶ É uma das estruturas mais básicas para promover o reuso

```
<?php
//exemplo de função

function nome_da_funcao($arg1, $arg2, $arg3){
    $valor = $arg1 + $arg2 + $arg3;
    return $valor;
}
?>
```

Funções

Exemplo

► Calcular o IMC

```
<?php
    function calcula_IMC ($peso, $altura) {
        return $peso / ($altura * $altura);
    }

    echo calcula_IMC(70, 1.85);
?>
```

Funções

Passagem de parâmetros - por valor

- ▶ É possível fazer a passagem de parâmetros de duas formas
 - ▶ Por valor (*by value*)
 - ▶ Por referência (*by reference*)
- ▶ Por padrão, os parâmetros são por valor
 - ▶ O parâmetro que a função recebe é tratado como local
 - ▶ Não altera seu valor externo

```
<?php
//exemplo de passagem por valor
function incrementa($var, $valor) {
    $var += $valor;
}

$a = 10;
incrementa($a, 20);
echo $a;
?>
```



Resultado:

10

Funções

Passagem de parâmetros - por referência

- ▶ Para realizar a passagem por referência, basta utilizar o operador & na frente do parâmetro
- ▶ As modificações realizadas no conteúdo da variável são válidas no contexto externo à função

```
<?php
//exemplo de passagem por referência
function incrementa(&$var, $valor) {
    $var += $valor;
}

$a = 10;
incrementa($a, 20);
echo $a;
?>
```



Resultado:

30

Funções

Passagem de parâmetros - valor padrão

- ▶ É possível definir valores padrão para parâmetros
- ▶ Se a função for chamada sem especificar o parâmetro, o valor padrão será assumido

```
<?php
//exemplo de passagem por referência com valor padrão
function incrementa(&$var, $valor = 40) {
    $var += $valor;
}

$a = 10;
incrementa($a);
echo $a;
?>
```



Resultado:

50

Funções

Recursão

- ▶ É possível criar funções recursivas
- ▶ A função a seguir calcula o fatorial de um número

```
<?php
function fatorial($numero) {
    if($numero == 1){
        return $numero;
    }else{
        return $numero * fatorial($numero - 1);
    }
}

echo fatorial(5) . "<br>";
echo fatorial(7) . "<br>";
?>
```



Resultado:

120

5040

Observação

Para cada um dos exercícios, fornecer a função e pelo menos 3 testes de seu uso.

Exercícios 18, 19 e 20

- 18) Crie uma função que receba um número inteiro e verifique e retorne se ele é par ou ímpar

Salve seu arquivo como `ex18.php`

19. Crie uma função que receba dois números e verifique e retorne o maior

Salve seu arquivo como `ex19.php`

- 20) Crie uma função que receba o salário de uma pessoa, calcule e retorne o valor do imposto de renda relativo devido conforme a tabela abaixo

Base de cálculo mensal em R\$	Alíquota %	Parcela a deduzir do imposto em R\$
Até 1.787,77	<i>isento</i>	<i>isento</i>
De 1.787,78 até 2.679,29	7,5	134,08
De 2.679,30 até 3.572,43	15,0	335,03
De 3.572,44 até 4.463,81	22,5	602,96
Acima de 4.463,81	27,5	826,15

Exemplo para um salário de R\$ 3000:

$$(3000 * 15\%) - 335,03 = 114,97$$

Salve seu arquivo como `ex20.php`

Observação 2

Tecnicamente, o cálculo do Imposto de Renda Retido na Fonte não é exatamente assim, mas iremos considerar conforme especificado para fins didáticos.

Requisição de arquivos

- ▶ Em linguagens de script como o PHP, frequentemente precisamos incluir dentro de nossos programas
 - ▶ Arquivos com definições de funções
 - ▶ Constantes
 - ▶ Configurações
 - ▶ Definição de uma classe
- ▶ Para esta finalidade, podemos utilizar as seguintes funções do PHP:
 - ▶ `include`
 - ▶ `require`
 - ▶ `include_once`
 - ▶ `require_once`

Requisição de arquivos

include

- ▶ Inclui e avalia o arquivo informado
- ▶ Seu código (variáveis, objetos, etc...) tornam-se disponíveis a partir da linha que ocorre a inclusão
- ▶ Se o arquivo não existir, produzirá uma mensagem de advertência (*warning*)

biblioteca.php

```
<?php
function quadrado($numero) {
    return $numero * $numero;
}
?>
```

teste.php

```
<?php
//carrega o arquivo com a
//função necessária
include 'biblioteca.php';

//imprime o quadrado do num 4

echo quadrado(4);
?>
```

Requisição de arquivos

require

- ▶ Idêntico ao `include`
- ▶ Difere somente na manipulação de erros
- ▶ Se o arquivo não existir, produzirá uma mensagem de erro fatal (*fatal error*), encerrando o programa imediatamente

biblioteca.php

```
<?php
function quadrado($numero) {
    return $numero * $numero;
}
?>
```

teste.php

```
<?php
//carrega o arquivo com a
//função necessária
require 'biblioteca.php';

//imprime o quadrado do num 4

echo quadrado(4);
?>
```

Pesquisa

Pesquise sobre os principais tipos de erro do PHP:

- Notice
- Warning
- Error
 - Fatal
 - Parse

Submeta sua resposta pelo Moodle

Requisição de arquivos

`include_once` e `require_once`

- ▶ Funciona de maneira similar às funções `include` e `require`
- ▶ A diferença é que se o arquivo já tiver sido incluído anteriormente, ele não será incluído de novo

`imprime_oi.php`

```
<?php
    echo "Oi!";
?>
```

`teste.php`

```
<?php
    include_once 'imprime_oi.php';
    include_once 'imprime_oi.php';
    include_once 'imprime_oi.php';

    //imprimirá a mensagem "Oi!"
    //apenas uma vez
?>
```

Exercício 21

21) Copie as funções criadas nos exercícios 18, 19 e 20 para um arquivo chamado `biblioteca.inc.php` e inclua-o no arquivo `ex21.php`, fazendo em seguida pelo menos três chamadas de suas funções.

Dica

Experimente as funções `include` e `require` e modifique o nome de arquivo a ser incluído para um nome não-existente. Analise os erros gerados pelo PHP.

Funções internas

- ▶ O PHP oferece uma série de funções internas, embutidas em seu código, para tratamento de números, *strings*, *arrays*, etc...
- ▶ A seguir, estudaremos algumas delas
- ▶ É essencial a leitura do manual do PHP para aprender a lidar com as funções e, também, para procurar por funções não mencionadas aqui
- ▶ A lista completa das funções do PHP pode ser obtida em: http://php.net/manual/pt_BR/funcref.php

Funções para variáveis

- isset

Retorna `TRUE` se a variável foi iniciada (mesmo que seja uma *string* vazia ou o valor zero) e `FALSE` caso contrário

- ▶ Exemplo

```
<?php
    $var = '';

    // Será interpretado como TRUE imprimindo o texto
    if (isset($var)) {
        echo "Essa variável existe.";
    }
?>
```

Funções para variáveis (cont.)

- empty

Retorna TRUE se a variável possui conteúdo vazio (uma *string* vazia, o valor zero ou NULL) e FALSE caso contrário

- ▶ Exemplo

```
<?php
    $var = 0;

    //Retornará TRUE, porque $var é um valor vazio
    if (empty($var)) {
        echo 'o conteúdo de $var é: 0, vazio ou NULL';
    }

    //Retornará TRUE porque $var está 'setado'
    if (isset($var)) {
        echo '$var está "setado", apesar de vazio';
    }
?>
```


Funções para variáveis (cont.)

- unset

Destrói uma variável, ou seja, libera a memória ocupada por ela, fazendo com que ela deixe de existir

- ▶ Exemplo

```
<?php
    $var = 'eu existo';

    // Será interpretado como TRUE imprimindo o texto
    if (isset($var)) {
        echo "Essa variável existe.";
    }

    unset($var);

    // Será FALSE, pois a variável $var foi destruída
    if (isset($var)) {
        echo "Será que ela continua existindo?";
    }
?>
```

Funções para datas

- date

Retorna a data e/ou hora: atual do servidor ou do *timestamp* informado

Possui diversos argumentos, que podem ser utilizados de acordo com o formato da data e/ou hora desejado

► Exemplo - Data no formato 31/12/2015

```
<?php
    $data = date("d/m/Y");
    echo "Hoje é: $data";
?>
```

► Exemplo - Data no formato 31/12/2015 22:05:00

```
<?php
    $data = date("d/m/Y H:i:s");
    echo "Hoje é: $data";
?>
```

Formato	Descrição
d	Dia do mês com 2 dígitos
D	Representação textual do dia
m	Representação numérica do mês
M	Representação textual do mês
Y	Representação do ano com 2 dígitos
Y	Representação do ano com 4 dígitos
l ('L' minúsculo)	Descrição do dia da semana
h	Formato em 12 horas
H	Formato em 24 horas
i	Minutos
s	Segundos

Funções para datas (*continuação*)

- time

Retorna o *timestamp* Unix atual, ou seja, a quantidade de segundos desde 1º de Janeiro de 1970

Toma por base o relógio do servidor onde o PHP é executado

- ▶ Exemplo

```
<?php
    $timestamp = time();
    echo "Fazem $timestamp segundos desde 01/01/1970";
?>
```

Funções para datas (*continuação*)

- mktime

Retorna o *timestamp* Unix para a data especificada

- ▶ **Formato**

`mktime(hora, minuto, segundo, mês, dia, ano)`

- ▶ **Exemplo**

```
<?php
//Gera o timestamp da data 09/02/1993 as 20:03:58
$timestamp = mktime(20, 3, 58, 2, 9, 1993);

// Exibe: 09/02 ~ 20:03
echo date('d/m ~ H:i', $timestamp);
?>
```

Funções para datas (*continuação*)

- strtotime

Analisa a descrição em inglês e retorna o *timestamp* Unix

- ▶ Exemplo

```
<?php
//timestamp atual
echo strtotime("now");

//timestamp de 10 de setembro de 2000
echo strtotime("10 September 2000");

//timestamp de amanhã (daqui a 24h)
echo strtotime("+1 day");

//timestamp para daqui a uma semana, 2 dias, 4 horas e 2 segundos
echo strtotime("+1 week 2 days 4 hours 2 seconds");

//timestamp para a próxima quinta-feira
echo strtotime("next Thursday");

//timestamp da última segunda-feira
echo strtotime("last Monday");
?>
```

Funções para datas (*continuação*)

- checkdate

Retorna TRUE caso a data informada seja válida ou FALSE, caso contrário

- ▶ **Formato**

`checkdate(mês, dia, ano)`

- ▶ **Exemplo**

```
<?php
    $dia = 29;
    $mes = 2;
    $ano = 2015;

    if(checkdate($mes, $dia, $ano)){
        echo "Data válida";
    }else{
        echo "Data inválida";
    }
?>
```

Exercício 22

22) Incremente o arquivo `biblioteca.inc.php`, criado no exercício 21, incluindo:

- a) Uma função para verificar se uma data é válida, no formato:

```
verifica_data(dia, mês, ano)
```

Dica: utilize a função interna `checkdate`

- b) Uma função para receber um timestamp e retornar sua data equivalente por extenso, no formato:

```
data_extenso_timestamp(timestamp)
```

Exemplo:

```
data_extenso_timestamp(1451516400);  
//imprime: 31 de Dezembro de 2015
```

- c) Uma função para receber uma data e retorná-la escrita por extenso, no formato:

```
data_extenso(dia, mês, ano)
```

Exemplo:

```
data_extenso(31, 12, 2015);  
//imprime: 31 de Dezembro de 2015
```

Dica: utilize `mktime` e a função criada anteriormente (letra B)

Importante para letra C

Verificar primeiramente se a data informada é válida.

Dica: utilize a função criada anteriormente (letra A).

Inclua o arquivo `biblioteca.inc.php` no arquivo `ex22.php` e faça pelo menos três chamadas das funções recentemente criadas

Funções para números

- rand

Gera um número inteiro aleatório

- ▶ Formato

`rand(mínimo, máximo)`

- ▶ Exemplo

```
<?php
//gera um número aleatório entre 0 e 1000
echo rand(0, 1000);
?>
```


Funções para números (cont.)

- number format

Formata um número para estilo moeda

- ▶ **Formato**

```
number_format(numero, qtd_casas_decimais, sep_decimal, sep_milhar)
```

- ▶ **Exemplo**

```
<?php
$valor = 1024.33;

//formata para R$ 1.024,33
echo 'R$ ' . number_format($valor, 2, ',', '.');

//formata para US$ 1,024.33
echo 'US$ ' . number_format($valor, 2, '.', ',');
?>
```

Funções matemáticas

- pow

Potência

- ▶ Formato

`pow(base, expoente)`

- ▶ Exemplo

```
<?php
//calcula 4 elevado a 2 = 16
echo pow(4, 2);
?>
```

- sqrt

Raiz quadrada

- ▶ Formato

`sqrt(numero)`

- ▶ Exemplo

```
<?php
//calcula a raiz
//quadrada de 81 = 9
echo sqrt(81);
?>
```

Funções matemáticas *(continuação)*

- ceil

Arredonda para cima

- ▶ Formato

`ceil(numero)`

- ▶ Exemplo

```
<?php
echo ceil(4.3);      // 5
echo ceil(9.999);    // 10
echo ceil(-3.14);    // -3
?>
```

- floor

Arredonda para baixo

- ▶ Formato

`floor(numero)`

- ▶ Exemplo

```
<?php
echo floor(4.3);     // 4
echo floor(9.999);   // 9
echo floor(-3.14);   // -4
?>
```

Exercício 23

23) Incremente o arquivo `biblioteca.inc.php`, incluindo:

- a) Uma função para gerar um número aleatório com X casas decimais:

`gera_decimal_aleatorio(mínimo, máximo, qtd_casas_decimais)`

Exemplo:

```
gera_decimal_aleatorio(0, 1000, 2);  
//imprime: 435.92
```

Dica: utilize a função interna `rand`

- b) Uma função para formatar um dado número em Reais:

`formata_reais(numero)`

Exemplo:

```
formata_reais(10123.45);  
//imprime: R$ 10.123,45
```

Dica: utilize a função interna `number_format`

Importante para letra A

- i) Caso não seja informado, considerar o valor 2 para a quantidade de casas decimais.
- ii) Considere o primeiro parâmetro sempre menor que o segundo.
- iii) Considere o terceiro parâmetro um número inteiro positivo diferente de 0.

Inclua o arquivo `biblioteca.inc.php` no arquivo `ex23.php` e faça pelo menos três chamadas das funções recentemente criadas

Funções para *strings*

○ strtoupper

Converte para maiúsculas

► Formato

```
strtoupper(string)
```

► Exemplo

```
<?php
//imprime MAIÚSCULO
echo strtoupper('maíúsculo');
?>
```

No exemplo anterior, a letra "ú" permanece em minúsculo. Para funcionar com letras acentuadas, usar mb strtoupper

```
<?php
//imprime MAIÚSCULO
echo mb_strtoupper('maíúsculo', 'utf-8');
?>
```

○ strtolower

Converte para minúsculas

► Formato

```
strtolower(string)
```

► Exemplo

```
<?php
//imprime minúsculo
echo strtolower('MINÚSCULO');
?>
```

No exemplo anterior, a letra "Ú" permanece em maiúsculo. Para funcionar com letras acentuadas, usar mb strtolower

```
<?php
//imprime minúsculo
echo mb_strtolower('MINÚSCULO', 'utf-8');
?>
```

Funções para *strings* (continuação)

○ substr

Retorna uma parte de uma *string*

► Formato

```
substr(string, começo[, tam])
```

► Exemplo

```
<?php
//retorna "f"
$rest = substr("abcdef", -1);

//retorna "ef"
$rest = substr("abcdef", -2);

//retorna "d"
$rest = substr("abcdef", -3, 1);
?>
```

○ str_replace

Substitui todas ocorrências

► Formato

```
str_replace(procura, substitui, texto)
```

► Exemplo

```
<?php
$txt = 'O Rato roeu a roupa';

//substitui Rato por Leão
echo str_replace('Rato', 'Leão', $txt);
?>
```

Funções para *strings* (continuação)

○ strpos

Encontra a posição da primeira ocorrência de uma *string*

► Formato

```
strpos(string, procurar_por)
```

► Exemplo

```
<?php
$txt = 'O Rato roeu a roupa';
$encontrar = 'roupa';

//informa a posição de 'roupa' (14)
echo "Posição de $encontrar: " .
strpos($txt, $encontrar);
?>
```

○ trim

Retira espaço no início e final de uma *string*

► Formato

```
trim(string)
```

► Exemplo

```
<?php
$str = '    olá mundo    ';

echo trim($str);

echo strlen(trim($str));
echo strlen($str);
?>
```

Exercício 24

24) Incremente o arquivo `biblioteca.inc.php`, incluindo:

- a) Uma função para transformar todas as vogais da *string* fornecida por '?':

`oculta_vogais(string)`

Exemplo:

```
oculta_vogais("IFGoiano");  
  
//imprime: ?FG???n?
```

- b) Uma função para retornar uma *string* em ordem inversa:

`inverte(string)`

Exemplo:

```
inverte("Oi, pessoal!");  
  
//imprime: !laossep ,iO
```

Importante: não vale usar a função interna `strrev`

Inclua o arquivo `biblioteca.inc.php` no arquivo `ex24.php` e faça pelo menos três chamadas das funções recentemente criadas

Arrays

- ▶ *Arrays* (ou vetores) podem ser usados para armazenar
 - ▶ números
 - ▶ *strings*
 - ▶ objetos
 - ▶ outros *arrays*
 - ▶ etc...
- ▶ O PHP oferece uma série de funções internas úteis para o tratamento de *arrays*

Arrays

Criando um *array*

- ▶ *Arrays* são acessados por meio de sua posição (índice)
- ▶ Para criar um *array*, pode-se utilizar a função `array()`

```
$cores = array('vermelho', 'azul', 'verde');
```

ou

```
$cores = array(0=>'vermelho', 1=>'azul', 2=>'verde');
```

- ▶ Outra forma é adicionar valores diretamente com a sintaxe:

```
$nomes[] = 'maria';  
$nomes[] = 'joão';  
$nomes[] = 'carlos';  
$nomes[] = 'josé';
```

Arrays

Criando um *array* (continuação)

- De qualquer forma, para acessar o *array*, basta indicar seu índice entre colchetes:

```
echo $cores[0];    //resultado = vermelho
echo $cores[1];    //resultado = azul
echo $cores[2];    //resultado = verde
```

```
echo $nomes[0];    //resultado = maria
echo $nomes[1];    //resultado = joão
echo $nomes[2];    //resultado = carlos
echo $nomes[3];    //resultado = josé
```

Arrays

Arrays associativos

- ▶ Além de poder utilizar índices numéricos, também é possível usar uma *string* como chave de uma posição do *array*

```
$cores = array('vermelho' => '#FF0000', 'azul' => '#0000FF', 'verde' => '#00FF00');
```

Para facilitar a leitura, pode-se utilizar quebras de linha e espaços adicionais:

```
$cores = array('vermelho' => '#FF0000',  
              'azul'      => '#0000FF',  
              'verde'     => '#00FF00');
```

- ▶ Outra forma é simplesmente adicionar valores, com a sintaxe:

```
$pessoa['nome'] = 'Maria da Silva';  
$pessoa['rua']  = 'São João';  
$pessoa['bairro'] = 'Cidade Alta';  
$pessoa['cidade'] = 'Porto Alegre';
```

Arrays

Arrays associativos (continuação)

- De qualquer forma, para acessar o *array*, basta indicar sua chave entre colchetes:

```
echo $cores['vermelho']; //resultado = #FF0000
echo $cores['azul'];     //resultado = #0000FF
echo $cores['verde'];    //resultado = #00FF00
```

```
echo $pessoa['nome'];    //resultado = Maria da Silva
echo $pessoa['rua'];     //resultado = São João
echo $pessoa['bairro'];  //resultado = Cidade Alta
echo $pessoa['cidade'];  //resultado = Porto Alegre
```

Arrays

Iterações

- Os *arrays* podem ser iterados pelo operador FOREACH

```
<?php
    $frutas['cor']      = 'vermelha';
    $frutas['sabor']    = 'doce';
    $frutas['formato']  = 'redonda';
    $frutas['nome']     = 'maçã';

    foreach($frutas as $chave => $valor){
        echo "$chave => $valor<br>";
    }
?>
```



Resultado:

```
cor => vermelha
sabor => doce
formato => redonda
nome => maçã
```

Arrays

Acesso

- As posições de um *array* podem ser acessadas a qualquer momento e sobre elas operações podem ser realizadas

```
<?php
    $minha_multa['carro'] = 'Pálio';
    $minha_multa['valor'] = 178.00;

    //alteração de valores
    $minha_multa['carro'] .= ' ED 1.0';
    $minha_multa['valor'] += 20;

    //exibe o array
    var_dump($minha_multa);
?>
```



Resultado:

```
array(2) {
    ["carro"]=>
    string(12) "Pálio ED 1.0"
    ["valor"]=>
    float(198)
}
```

Dica

Através da função `print_r` ou `var_dump` é possível imprimir toda a estrutura de um *array*.

Arrays

Arrays multidimensionais

- Arrays multidimensionais ou matrizes são *arrays* nos quais suas posições podem conter outros *arrays*

```
<?php
$carros = array('Palio' => array('cor' => 'azul',
                                'potência' => '1.0',
                                'opcionais' => 'Ar Cond.'),
               'Corsa' => array('cor' => 'cinza',
                                'potência' => '1.3',
                                'opcionais' => 'MP3'),
               'Gol'   => array('cor' => 'branco',
                                'potência' => '1.0',
                                'opcionais' => 'Metálica')
               );

echo $carros['Palio']['opcionais']; //resultado = Ar Cond.
?>
```


Arrays

Arrays multidimensionais (continuação)

- Outra forma de criar um *array* multidimensional é simplesmente atribuindo-lhe valores

```
<?php
$carros['Palio']['cor']          = 'azul';
$carros['Palio']['potência']    = '1.0';
$carros['Palio']['opcionais']   = 'Ar Cond.';

$carros['Corsa']['cor']         = 'cinza';
$carros['Corsa']['potência']    = '1.3';
$carros['Corsa']['opcionais']   = 'MP3';

$carros['Gol']['cor']           = 'branco';
$carros['Gol']['potência']      = '1.0';
$carros['Gol']['opcionais']     = 'Metálica';

echo $carros['Palio']['opcionais']; //resultado = Ar Cond.
?>
```

Arrays

Arrays multidimensionais (continuação)

- Para realizar iterações em um *array* multidimensional, é preciso observar quantos níveis ele possui

```
<?php
//...considerando que os arrays anteriores já estão no código

foreach($carros as $modelo => $caracts){
    echo "=> modelo $modelo<br>";

    foreach($caracts as $caract => $valor){
        echo "característica $caract => $valor<br>";
    }
}
?>
```

Resultado:

```
=> modelo Palio
característica cor => azul
característica potência => 1.0
característica opcionais => Ar Cond.
=> modelo Corsa
característica cor => cinza
característica potência => 1.3
característica opcionais => MP3
=> modelo Gol
característica cor => branco
característica potência => 1.0
característica opcionais => Metalica
```

Importante

Usar um laço de repetição e não `print_r` ou `var_dump` para imprimir o conteúdo dos arrays.

Exercícios 25, 26 e 27

- 25) Crie um *array* de 100 posições utilizando um laço de repetição (FOR, por exemplo), preenchendo cada posição com números aleatórios de 0 a 1000 (função `rand`). Em seguida, imprima o conteúdo do *array* utilizando um laço de repetição (FOREACH, por exemplo).

Salve seu arquivo como `ex25.php`

- 26) Crie um *array* chamado `$alunos` e utilize-o para armazenar o nome de pelo menos 5 alunos. Em seguida, altere o nome dos três primeiros alunos e, então, imprima o conteúdo do *array* utilizando um laço de repetição (FOREACH, por exemplo).

Salve seu arquivo como `ex26.php`

- 27) Crie um *array* multidimensional chamado `$clientes` e, em cada posição deste *array*, armazene nome, CPF, RG e telefone de pelo menos 5 clientes. Em seguida, imprima o conteúdo do *array* utilizando laços de repetição (FOREACH, por exemplo).

Salve seu arquivo como `ex27.php`

Funções para *arrays*

○ array push

Adiciona um ou mais elementos ao final de um *array*

Tem o mesmo efeito de utilizar a sintaxe: `$array[] = $valor`

► Exemplo

```
<?php
    $a = array('verde', 'azul',
'vermelho');

    array_push($a, 'amarelo');
```

```
var_dump($a);
?>
```



Resultado:

```
array(4) {
    [0]=> string(5) "verde"
    [1]=> string(4) "azul"
    [2]=> string(8) "vermelho"
    [3]=> string(7) "amarelo"
}
```

○ array pop

Retira um elemento do final de um *array*

► Exemplo

```
<?php
    $a = array('verde', 'azul',
'vermelho');

    array_pop($a);

    var_dump($a);
?>
```



Resultado:

```
array(2) {
    [0]=> string(5) "verde"
    [1]=> string(4) "azul"
}
```



Funções para *arrays* (continuação)

○ array shift

Retira um elemento do início de um *array*

► Exemplo

```
<?php
$a = array('verde', 'azul',
'vermelho', 'amarelo');

array_shift($a);

var_dump($a);
?>
```



Resultado:

```
array(3) {
  [0]=> string(4) "azul"
  [1]=> string(8) "vermelho"
  [2]=> string(7) "amarelo"
}
```



INSTITUTO FEDERAL
GOIANO

○ array unshift

Adiciona um ou mais elementos do início de um *array*

► Exemplo

```
<?php
$a = array('verde', 'azul',
'vermelho');

array_unshift($a, 'amarelo');

var_dump($a);
?>
```



Resultado:

```
array(4) {
  [0]=> string(7) "amarelo"
  [1]=> string(5) "verde"
  [2]=> string(4) "azul"
  [3]=> string(8) "vermelho"
}
```

Funções para *arrays* (continuação)

○ array reverse

Retorna um *array* na ordem inversa

► Exemplo

```
<?php
$a[0] = 'green';
$a[1] = 'yellow';
$a[2] = 'red';
$a[3] = 'blue';

$b = array_reverse($a);

var_dump($b);
?>
```



Resultado:

```
array(4) {
    [3]=> string(4) "blue"
    [2]=> string(3) "red"
    [1]=> string(6) "yellow"
    [0]=> string(5) "green"
}
```



GOIANO

Prof. - Hypertext Preprocessor
Prof. Marcos Alves Vieira
marcos.vieira@ifgoiano.edu.br

○ array merge

Mescla dois ou mais *arrays*

Obs: se dois *arrays* tiverem conteúdo indexado pela mesma chave (*string*), o segundo sobrepõe o primeiro

► Exemplo

```
<?php
$a = array('verde', 'azul');
$b = array('vermelho',
'amarelo');
$c = array_merge($a, $b);

var_dump($c);
?>
```



Resultado:

```
array(4) {
    [0]=> string(5) "verde"
    [1]=> string(4) "azul"
    [2]=> string(8) "vermelho"
    [3]=> string(7) "amarelo"
}
```

Funções para *arrays* (continuação)

- in_array

Verifica se um valor existe em um *array*

- ▶ Exemplo

```
<?php
$a[] = 'refrigerante';
$a[] = 'cerveja';
$a[] = 'vodka';
$a[] = 'suco natural';

if(in_array('suco natural', $a)){
    echo 'suco natural encontrado';
}
?>
```



Resultado:

suco natural encontrado

- sort

Ordena um *array* pelos valores

- ▶ Exemplo

```
<?php
$a[] = 'refrigerante';
$a[] = 'cerveja';
$a[] = 'vodka';
$a[] = 'suco natural';

sort($a);

print_r($a);
?>
```



Resultado:

Array

```
(
    [0] => cerveja
    [1] => refrigerante
    [2] => suco natural
    [3] => vodka
)
```

Veja também as funções de ordenação:

- ▶ rsort
- ▶ asort
- ▶ arsort
- ▶ ksort
- ▶ krsort



Funções para *arrays* (continuação)

○ count

Retorna a quantidade de elementos de um *array*

► Exemplo

```
<?php
$a[] = 'refrigerante';
$a[] = 'cerveja';
$a[] = 'vodka';
$a[] = 'suco natural';

echo 'o array $a contém ' . count($a) . ' posições';
?>
```



Resultado:

o array \$a contém 4 posições

A função `count` também pode ser usada para iterar em *arrays*

```
<?php
//...considerando que o array anterior já está no código

for($i = 0; $i < count($a); $i++){
    echo $a[$i] . '<br>';
}
?>
```



Resultado:

refrigerante
cerveja
vodka
suco natural

Funções para *arrays* (continuação)

○ explode

Converte uma *string* em um *array*, separando os elementos por meio de um separador

► Exemplo

```
<?php
$string = '31/12/2004';
$array = explode('/', $string);

var_dump($array);
?>
```



Resultado:

```
array(3) {
  [0]=>
  string(2) "31"
  [1]=>
  string(2) "12"
  [2]=>
  string(4) "2004"
}
```



○ implode

Converte um *array* em uma *string*, separando os elementos por meio de um separador

► Exemplo

```
<?php
$padrao[] = 'Maria';
$padrao[] = 'Paulo';
$padrao[] = 'José';

$resultado = implode(' + ', $padrao);

var_dump($resultado);
?>
```



Resultado:

```
string(20) "Maria + Paulo + José"
```

Exercícios 28, 29 e 30

- 28) Crie uma matriz 5x5 e insira valores inteiros aleatórios (use um laço de repetição, como o FOR, e a função `rand`). Depois, faça uma função que receba uma matriz de qualquer dimensão como parâmetro e retorne o maior valor encontrado na matriz.

Salve seu arquivo como `ex28.php`

- 29) Crie dois *arrays*, cada um com 50 posições preenchidas com números inteiros aleatórios (use um laço de repetição, como o FOR, e a função `rand`). Em seguida, mescle estes dois *arrays* em um terceiro *array* e, por fim, ordene o terceiro *array* e imprima-o utilizando um laço de repetição (como o FOREACH).

Salve seu arquivo como `ex29.php`

- 30) Crie uma função que verifique se um determinado valor existe em um *array* e retorne sua chave caso exista ou uma mensagem informando que o valor não existe no *array*.

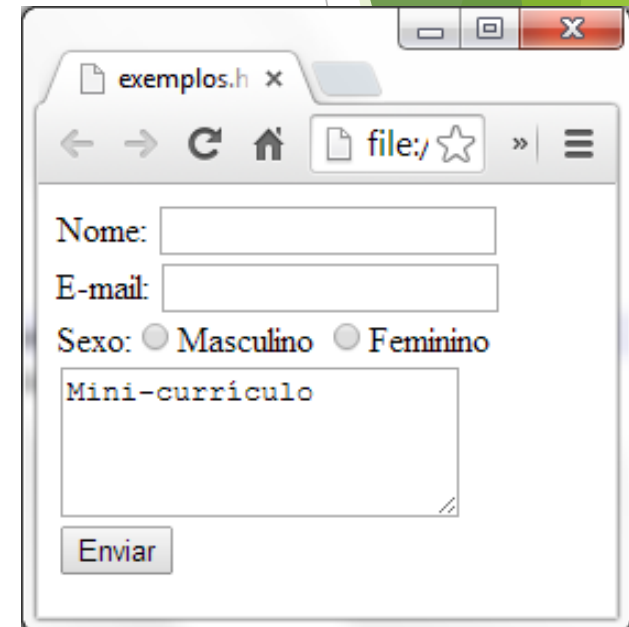
Dica: pesquise o funcionamento da função [array_search](#)

Salve seu arquivo como `ex30.php`

Formulários

Relembrando os conceitos

- ▶ Formulários são seções específicas de um documento projetados para
 - ▶ Coletar dados fornecidos pelo usuário
 - ▶ Enviá-los para serem processados por um *script*
- ▶ Os objetos que compõem um formulário são chamados de *controles*
- ▶ São exemplos de controles de um formulário
 - ▶ Botões
 - ▶ Entradas de texto
 - ▶ *Checkboxes*
 - ▶ Seleção de arquivos
 - ▶ *Radio buttons*
 - ▶ Controle oculto
 - ▶ Menus



A screenshot of a web browser window displaying a form. The browser's address bar shows 'file:/' and the tab is labeled 'exemplos.h'. The form contains the following elements: a text input field for 'Nome:', another for 'E-mail:', a 'Sexo:' label with two radio buttons labeled 'Masculino' and 'Feminino', a large text area labeled 'Mini-curriculo', and a button labeled 'Enviar'.

Formulários

Elementos de formulários - `form`

- ▶ O elemento `form` é o container para os controles do formulário
- ▶ Ele contém o *layout* composto pelos conteúdos do formulário
 - ▶ Pode conter textos e marcação HTML, além de seus controles
- ▶ Atributos principais:
 - ▶ `action` → define o endereço do programa (script) que irá processar o formulário após a sua submissão
 - ▶ `method` → define o método de envio dos dados (POST ou GET)

```
<form action="processa_contato.php" method="post">
..controles do formulário...
</form>
```

Formulários

Elementos de formulários - `input`

- O elemento `input` cria vários tipos de controle que são definidos pelo atributo `type`

Valor do atributo <code>type</code>	Controle criado
<code>hidden</code>	Envio oculto de dados
<code>text</code>	Para entrada de uma linha simples de texto
<code>password</code>	Semelhante ao anterior, mas os caracteres digitados não são legíveis (por exemplo, são substituídos por uma série de asteriscos)
<code>radio</code>	Radio button
<code>checkbox</code>	Checkbox
<code>file</code>	Seleção de arquivo para envio
<code>submit</code>	Botão de envio do formulário
<code>reset</code>	Botão para limpar os dados entrados pelo usuário
<code>button</code>	Botão associado a um tipo de processamento específico criado pelo autor
<code>image</code>	Botão gráfico de envio do formulário

Formulários

Elementos de formulários - input (continuação)

```
<form action="">
<input type="hidden" />

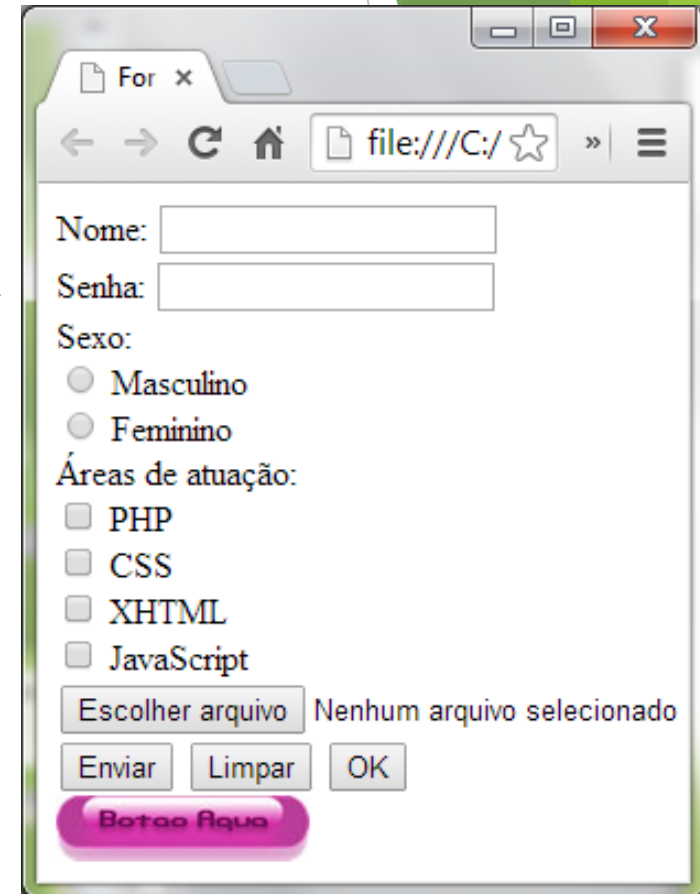
Nome: <input type="text" /><br />
Senha: <input type="password" /><br />

Sexo:<br />
<input type="radio" name="sexo" /> Masculino<br />
<input type="radio" name="sexo" /> Feminino<br />

Áreas de atuação:<br />
<input type="checkbox" /> PHP<br />
<input type="checkbox" /> CSS<br />
<input type="checkbox" /> XHTML<br />
<input type="checkbox" /> JavaScript<br />

<input type="file" /><br />

<input type="submit" value="Enviar" />
<input type="reset" value="Limpar" />
<input type="button" value="OK" /><br />
<input type="image" src="botao-aqua.jpg" />
</form>
```



Formulários

Elementos de formulários - `button`

- ▶ O elemento `button` cria controles do tipo botão de maneira semelhante ao elemento `input`
- ▶ A diferença é que o elemento `input` é vazio
 - ▶ Podemos inserir conteúdo usando o elemento `button`
- ▶ O tipo de botão criado é definido pelo atributo `type`:

Tipo	Finalidade
<code>submit</code>	Botão de envio do formulário
<code>reset</code>	Botão para limpar os dados entrados pelo usuário
<code>button</code>	Botão associado a scripts programados pelo autor

Formulários

Elementos de formulários - button (continuação)

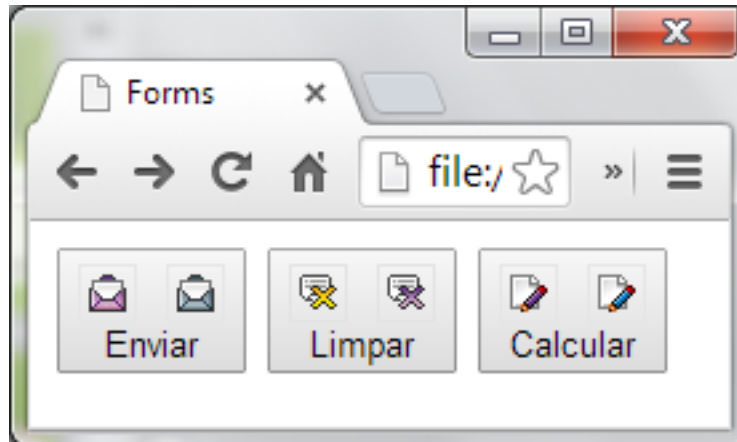
```
<form action="">
```

```
<button type="submit">  
    <br />Enviar  
</button>
```

```
<button type="reset">  
    <br />Limpar  
</button>
```

```
<button type="button">  
    <br />Calcular  
</button>
```

```
</form>
```



Formulários

Elementos de formulários - `select`, `option` e `optgroup`

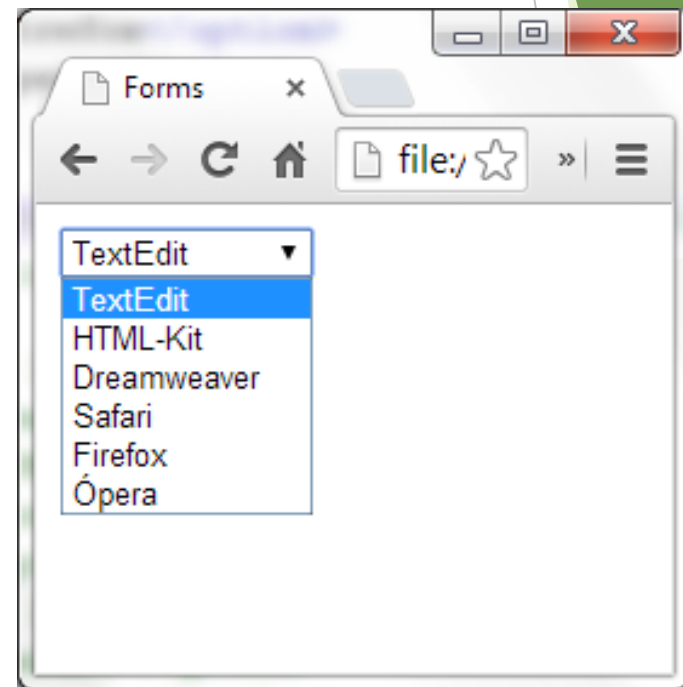
- ▶ O elemento `select` destina-se a criar um menu de escolhas em um formulário
- ▶ Cada item do menu é marcado com o elemento `option`
- ▶ Para agrupar itens do menu que tenham alguma relação entre si, usamos o elemento `optgroup`
 - ▶ Útil para fazer escolhas em uma lista muito extensa de opções
 - ▶ Agrupar opções de natureza idêntica
 - ▶ Facilita a localização das opções

Formulários

Elementos de formulários - select, option e optgroup (continuação)

► Sem usar optgroup

```
<form action="">
<select>
<option>TextEdit</option>
<option>HTML-Kit</option>
<option>Dreamweaver</option>
<option>Safari</option>
<option>Firefox</option>
<option>Ópera</option>
</select>
</form>
```

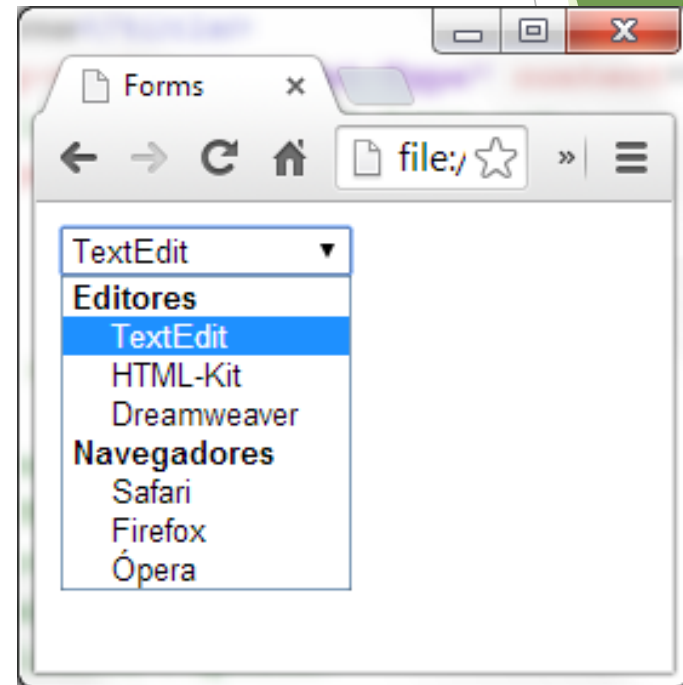


Formulários

Elementos de formulários - select, option e optgroup (continuação)

► Usando optgroup

```
<form action="">
<select>
<optgroup label="Editores">
<option>TextEdit</option>
<option>HTML-Kit</option>
<option>Dreamweaver</option>
</optgroup>
<optgroup label="Navegadores">
<option>Safari</option>
<option>Firefox</option>
<option>Ópera</option>
</optgroup>
</select>
```

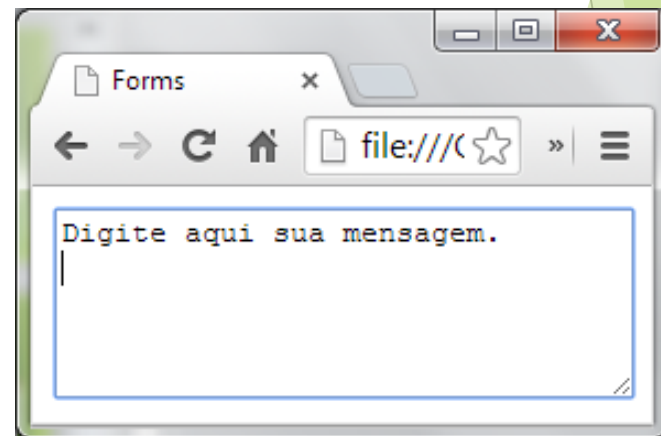


Formulários

Elementos de formulários - textarea

- ▶ O elemento `textarea` destina-se a criar um controle do tipo texto permitindo entradas do usuário em múltiplas linhas
- ▶ Atributos principais:
 - ▶ `rows` → define a quantidade de linhas do `textarea` (altura)
 - ▶ `cols` → define a quantidade de colunas do `textarea` (largura)

```
<form action="">  
<textarea rows="5" cols="30">  
Digite aqui sua mensagem.  
</textarea>  
</form>
```



Formulários

Elementos de formulários - `label`

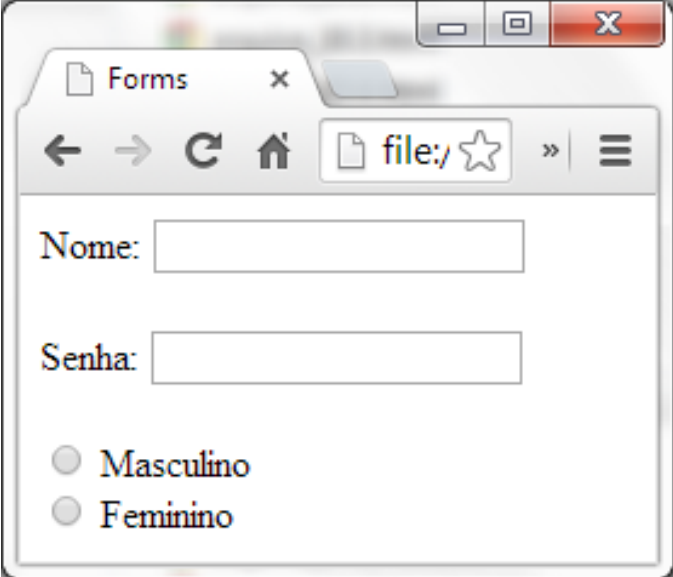
- ▶ A maioria dos controles de formulário não possuem rótulos, como campos de texto, *checkboxes*, *radio buttons* e menus.
- ▶ O elemento `label` é utilizado para especificar rótulos para controles
 - ▶ É um item importante para incrementar a acessibilidade (por exemplo, leitores de tela)
- ▶ Cada elemento `label` associa um rótulo a um (e somente um) controle no formulário, com a finalidade de agregar informação ao controle
- ▶ O atributo `for` é usado para associar um rótulo com o seu controle
 - ▶ O valor do atributo `for` deve ser o mesmo valor do atributo `id` do controle a ele associado

Formulários

Elementos de formulários - label (continuação)

► Associação explícita do elemento label

```
<form action="">
<label for="nome">Nome:</label>
<input type="text" id="nome" />
<br /><br />
<label for="senha">Senha:</label>
<input type="password" id="senha" />
<br /><br />
<input type="radio" id="masc" />
<label for="masc">Masculino</label>
<br />
<input type="radio" id="fem" />
<label for="fem">Feminino</label>
</form>
```



The screenshot shows a web browser window with a single tab titled 'Forms'. The address bar shows a local file path. The rendered form contains the following elements:

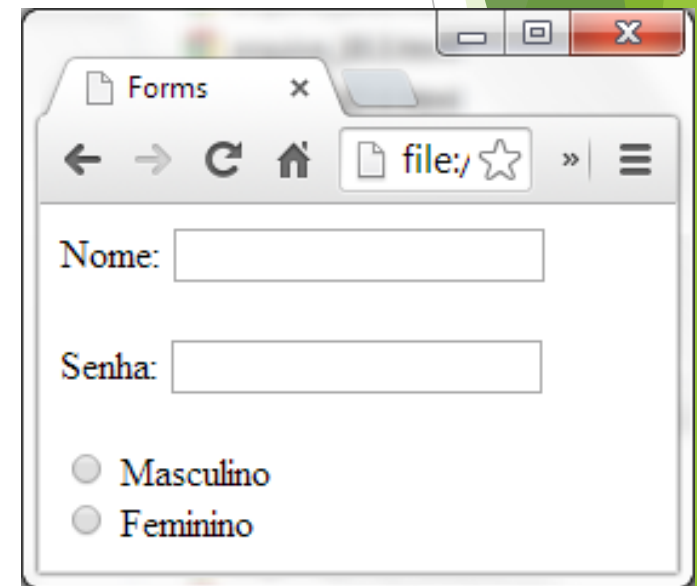
- A text input field preceded by the label 'Nome:'.
- A password input field preceded by the label 'Senha:'.
- Two radio buttons below the password field, labeled 'Masculino' and 'Feminino'.

Formulários

Elementos de formulários - label (continuação)

► Associação implícita do elemento label

```
<form action="">
<label>
    Nome:<input type="text" />
</label>
<br /><br />
<label>
    Senha:<input type="password" />
</label>
<br /><br />
<label>
    <input type="radio" />Masculino
</label>
<br />
<label>
    <input type="radio" />Feminino
</label>
</form>
```



Formulários

Elementos de formulários - `fieldset` e `legend`

- ▶ Têm a função de estruturar formulários
- ▶ O elemento `fieldset` destina-se a agrupar um conjunto de controles do formulário que tenham finalidades relacionadas
 - ▶ o conjunto de campos de texto destinados a coletar dados residenciais
 - ▶ o conjunto de *checkboxes* destinado a coletar preferências do usuário
- ▶ A cada `fieldset` pode ser associado um elemento `legend`, que se destina a identificar com um rótulo (ou título) o respectivo `fieldset`
 - ▶ O elemento `legend` é importante para incrementar a acessibilidade

Formulários

Elementos de formulários - fieldset e legend (continuação)

```
<form action="">
<fieldset>
<legend>Informações pessoais</legend><br />
<label>Nome: <input type="text" /></label><br />
<label>Endereço: <input type="text" />
</label><br /><br />
...outros controles coletando dados pessoais...
</fieldset>

<fieldset>
<legend>Histórico médico</legend><br />
<label><input type="checkbox" />
Alérgico</label><br />
<label><input type="checkbox" />
Tonteiras</label><br />
<label><input type="checkbox" />
Vacinação</label><br />
...outros controles para histórico médico...
</fieldset>
</form>
```

The screenshot shows a web browser window with a single tab titled "Forms". The address bar shows the file path "file:///C:/Users/21". The form contains two distinct sections, each enclosed in a box with a title bar. The first section is titled "Informações pessoais" and contains two text input fields labeled "Nome:" and "Endereço:". Below these fields is a legend box labeled "fieldset". The second section is titled "Histórico médico" and contains three checkboxes labeled "Alérgico", "Tonteiras", and "Vacinação". Below these checkboxes is another legend box labeled "fieldset". Arrows from the "legend" label in the code point to these legend boxes.

Formulários

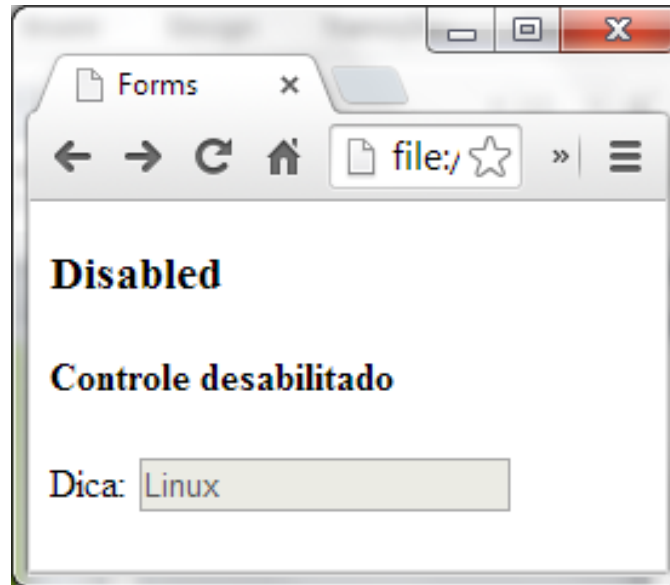
Atributos - `disabled` e `readonly`

- ▶ Pode ser necessário desabilitar a entrada de dados em um elemento ou colocá-lo como somente leitura
- ▶ O atributo `disabled` desabilita o controle para ações do usuário e produz os seguintes efeitos em um elemento:
 - ▶ Desabilita o controle para receber foco
 - ▶ Desabilita o controle na navegação por tabulação
 - ▶ Não envia o dado contido no controle para processamento quando o formulário for enviado
- ▶ Os seguintes elementos suportam o atributo `disabled`:
 - ▶ `button`
 - ▶ `input`
 - ▶ `optgroup`
 - ▶ `option`
 - ▶ `select`
 - ▶ `textarea`

Formulários

Atributos - disabled e readonly (cont.)

```
<form action="">
<label>
    Dica: <input disabled="disabled" value="Linux">
</label>
</form>
```



Formulários

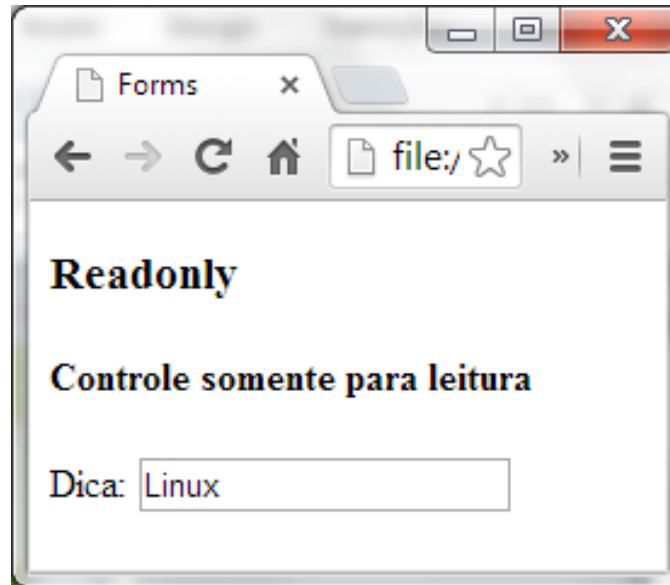
Atributos - `disabled` e `readonly`

- ▶ O atributo `readonly` põe o controle no modo somente para leitura e produz os seguintes efeitos em um elemento:
 - ▶ Recebe o foco, mas não pode ser modificado pelo usuário
 - ▶ São incluídos normalmente na navegação por tabulação
 - ▶ Pode enviar o dado contido no controle para processamento quando o formulário for enviado
- ▶ Os seguintes elementos suportam o atributo `readonly`:
 - ▶ `input`
 - ▶ `textarea`

Formulários

Atributos - disabled e readonly (cont.)

```
<form action="">
<label>
    Dica: <input readonly="readonly" value="Linux">
</label>
</form>
```



Formulários

Métodos de envio: POST e GET

- ▶ Os formulários podem ser submetidos para processamento por meio dos métodos `POST` ou `GET`

- ▶ Método `GET`

- ▶ os dados do formulário são anexados à URL (URI) do campo `action`, com um caractere '?' como separador
 - ▶ a URL resultante é enviada à página ou *script* que fará o processamento do formulário
 - ▶ os dados ficam armazenados no *array* associativo `$_GET`

- ▶ Método `POST`

- ▶ encapsula os dados do formulário dentro da mensagem que será enviada pelo protocolo HTTP à página ou *script* que fará o processamento do formulário
 - ▶ os dados ficam armazenados no *array* associativo `$_POST`

Os nomes das chaves do *array* equivalem aos nomes dos campos do formulário.

Mais informações, acesse:

<http://www.cs.tut.fi/~jkorpela/forms/methods.html>

Formulários

Qual método usar?

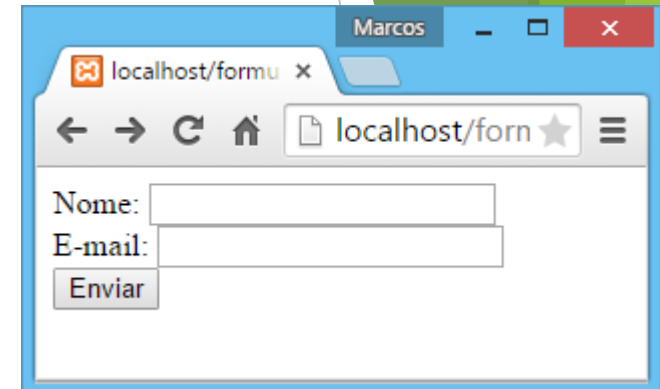
- ▶ O programador deve escolher o método mais adequado para cada caso, observando o seguinte:
 - ▶ O método `GET` pode ser utilizado sempre que o formulário possa ser enviado diversas vezes sem alterar o resultado obtido, ou seja, não causando efeitos colaterais
 - ▶ Exemplo: um campo de busca
 - ▶ O método `POST` deve ser utilizado quando enviamos uma quantidade maior de dados ou quando a submissão do formulário puder causar efeitos colaterais
 - ▶ Exemplo: gravação dos dados em um banco de dados
- ▶ Além disso, o método `POST` fornece uma certa segurança adicional, uma vez que os dados enviados não ficam visíveis na URL
 - ▶ Isso pode impedir o envio de dados inconsistentes por um usuário mal-intencionado

Formulários

Submetendo formulário via GET

formulario.php

```
1 <!DOCTYPE html>
2 <html>
3   <head></head>
4   <body>
5     <form action="processa.php" method="GET">
6       Nome: <input type="text" name="nome" />
7       <br />
8       E-mail: <input type="text" name="email" />
9       <br />
10      <input type="submit" value="Enviar" />
11    </form>
12  </body>
13 </html>
```

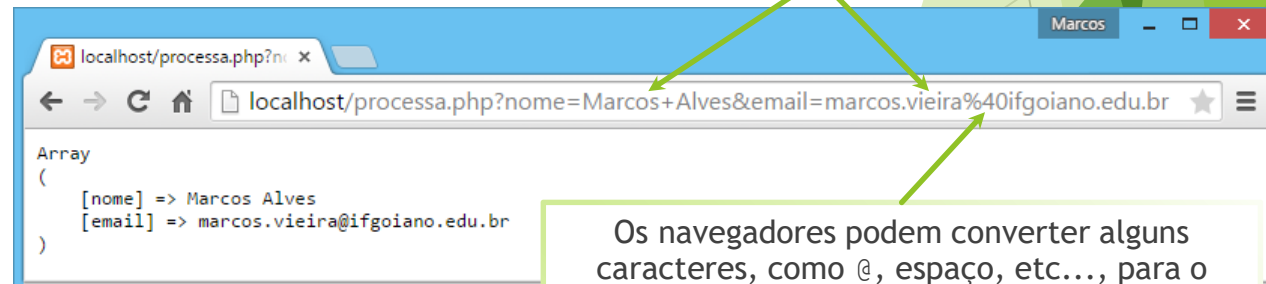


A screenshot of a web browser window titled 'Marcos'. The address bar shows 'localhost/formu'. The page content includes a form with two text input fields labeled 'Nome:' and 'E-mail:', and a button labeled 'Enviar'.

Note que os dados do formulário são passados na URL:
? -> representa o início da cadeia de variáveis
& -> identifica o início de uma nova variável
= -> separa as variáveis dos seus respectivos valores

processa.php

```
1 <?php
2   echo "<pre>";
3   print_r($_GET);
4   echo "</pre>";
5   ?>
```



A screenshot of a web browser window titled 'Marcos'. The address bar shows 'localhost/processa.php?nome=Marcos+Alves&email=marcos.vieira%40ifgoiano.edu.br'. The page content displays a preformatted array of the submitted data:

```
Array
(
    [nome] => Marcos Alves
    [email] => marcos.vieira@ifgoiano.edu.br
)
```

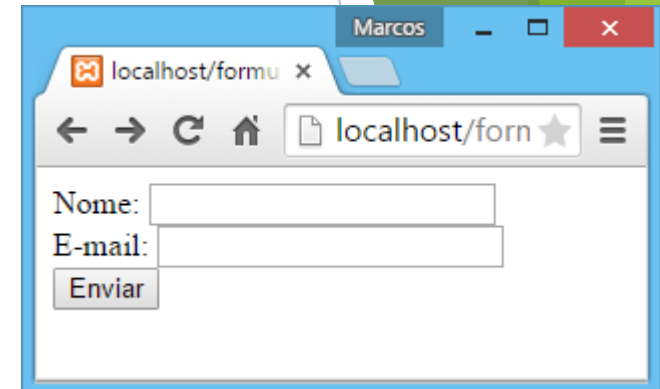
Os navegadores podem converter alguns caracteres, como @, espaço, etc..., para o valor hexadecimal de sua entidade HTML.
([mais informações](#))

Formulários

Submetendo formulário via POST

formulario.php

```
1 <!DOCTYPE html>
2 <html>
3   <head></head>
4   <body>
5     <form action="processa.php" method="POST">
6       Nome: <input type="text" name="nome" />
7       <br />
8       E-mail: <input type="text" name="email" />
9       <br />
10      <input type="submit" value="Enviar" />
11    </form>
12  </body>
13 </html>
```



Nome:

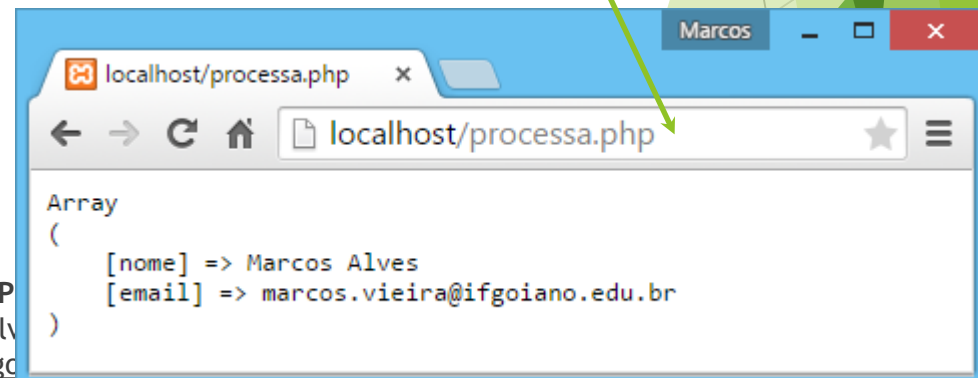
E-mail:

Enviar

Note que a URL não contém os dados do formulário.

processa.php

```
1 <?php
2   echo "<pre>";
3   print_r($_POST);
4   echo "</pre>";
5 <?>
```



localhost/processa.php

```
Array
(
    [nome] => Marcos Alves
    [email] => marcos.vieira@ifgoiano.edu.br
)
```

Exercício 31

31) Crie o formulário abaixo, salve-o como `meu_formulario.php` e aponte seu `action` para a página `exibe_dados.php`. Em seguida, crie a página `exibe_dados.php` e utilize-a para exibir os dados preenchidos no formulário.

Obs.: Crie uma versão de ambos arquivos com submissão de dados via `POST` e outra via `GET`.

`meu_formulario_post.php`

The screenshot shows a web browser window with the title 'Cadastro de Clientes'. The address bar shows 'localhost/meu_formulario_post.php'. The form is divided into three sections: 'Dados Pessoais', 'Endereço', and 'Observações'. The 'Dados Pessoais' section has fields for 'Nome', 'E-mail', 'Sexo' (with radio buttons for 'Masculino' and 'Feminino'), and 'Estado civil' (a dropdown menu). The 'Endereço' section has fields for 'Rua', 'Bairro', 'Cidade', 'Estado', and 'CEP'. The 'Observações' section is a large text area. At the bottom are 'Salvar' and 'Limpar' buttons. A green arrow points from the 'Opções' list to the 'Estado civil' dropdown menu.

Dados Pessoais

Nome:

E-mail:

Sexo: ☐ Masculino ☐ Feminino

Estado civil:

Endereço

Rua:

Bairro:

Cidade: Estado:

CEP:

Observações

Opções:

- Selecione...
- Casado
- Solteiro
- Divorciado
- Viúvo

`exibe_dados_post.php`

The screenshot shows a web browser window with the title 'localhost/exibe_dados_po'. The address bar shows 'localhost/exibe_dados_post.php'. The page displays the data from the form in a structured layout: 'Dados Pessoais' with fields for 'Nome', 'E-mail', 'Sexo', and 'Estado civil'; 'Endereço' with fields for 'Rua', 'Bairro', 'Cidade', 'Estado', and 'CEP'; and 'Observações'.

Dados Pessoais

Nome: Marcos Alves Vieira

E-mail: marcos.vieira@ifgoiano.edu.br

Sexo: masculino

Estado civil: casado

Endereço

Rua: Av. Pará, nº tal

Bairro: Centro

Cidade: Iporá Estado: GO

CEP: 76200-000

Observações

Sem observações no momento.

Exercício 32

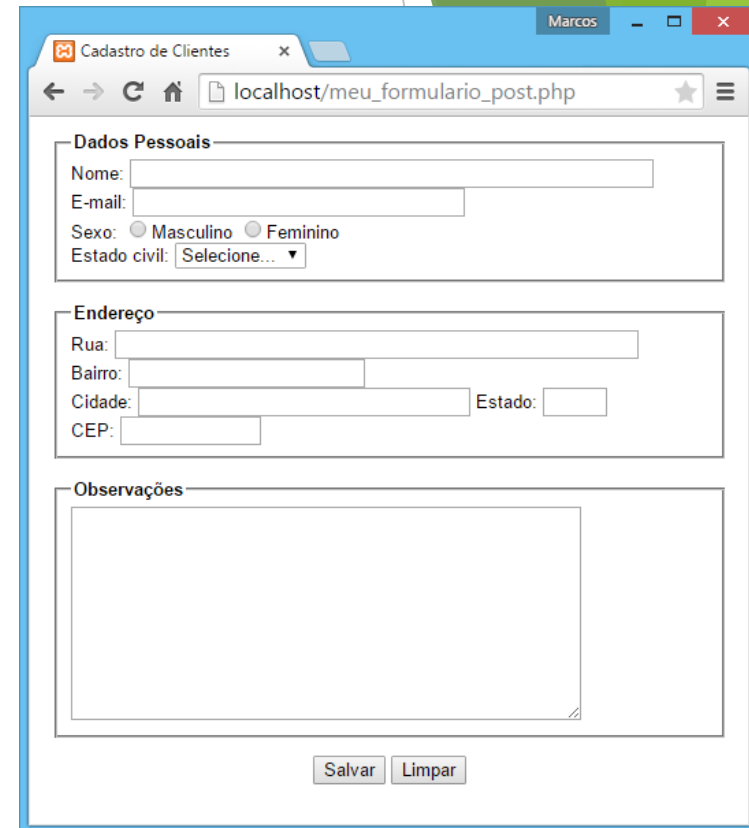
32) Modifique o arquivo `exibe_dados.php` (GET ou POST) criado no exercício anterior para validar os dados enviados pelo formulário antes de exibi-los na tela, conforme as seguintes regras:

- a) Todos os campos são obrigatórios, com exceção do campo "Observações"
- b) O campo "E-mail" deve conter um endereço de e-mail válido
- c) O campo "Estado civil" deve conter uma opção diferente de "Selecione..."
- d) O campo "Estado" deve conter duas letras e estas devem representar um estado brasileiro. Ex: GO, MG...
- e) O campo "CEP" deve conter um CEP válido, no formato xxxxx-xxx, onde x é um número de 0 a 9

IMPORTANTE:

Uma mensagem deve ser mostrada para o usuário para cada campo que não passar na validação. Apresente também uma opção para voltar ao formulário e editar os dados (*link*, botão, etc...). Exemplos:

```
<a href="javascript:history.back()">Voltar</a>  
<button onclick="history.back()">Voltar</button>
```



Publicando o seu site na Web

- ▶ Para que seu site possa ser acessado através de qualquer lugar do mundo, pela Internet, ele deve ter:
 - ▶ Domínio ou um endereço na Web
 - ▶ Domínio, subdomínio ou endereço oferecido pelo serviço de hospedagem
 - ▶ Hospedagem: estar hospedado em um servidor web acessível pela Internet
 - ▶ É o servidor físico onde os arquivos estarão armazenados
 - ▶ Nossos computadores com Apache instalado são servidores web, mas não podem ser acessados externamente, pela Internet
 - ▶ Mas isso pode ser resolvido :-)

Publicando o seu site na Web

Domínio

- ▶ Domínio é um nome que serve para localizar e identificar conjuntos de computadores na Internet
- ▶ Sem um domínio, cada site só poderia ser acessado pelo endereço IP do seu servidor web
 - ▶ Domínio: <http://www.uol.com.br>
 - ▶ Endereço IP: <http://200.147.67.142>
- ▶ A terminação do nome do domínio, também chamado de **domínio de topo** ou *top level domain* (TLD), indica o país onde aquele domínio foi registrado. [\(mais informações\)](#)
- ▶ O responsável pelo registro de domínios com terminação .br é o [Registro.br](#)
 - ▶ Atualmente, o registro de um domínio .br custa R\$ 40 por ano [\(fonte\)](#)

Publicando o seu site na Web

Hospedagem

- ▶ Ao registrar um domínio, você deve informar o endereço do servidor onde o seu site está hospedado. No servidor de hospedagem, você deve informar qual(is) domínio(s) aponta(m) para o seu site [\(mais informações\)](#)
- ▶ Existem vários serviços de hospedagem disponíveis, tanto pagos quanto gratuitos, oferecendo planos de acordo com as características do servidor
 - ▶ Sistema Operacional
 - ▶ Linguagens de programação suportadas
 - ▶ Tipo de SGBD (Sistema Gerenciador de Banco de Dados) e quantidade de bancos de dados que podem ser criados
 - ▶ Quantidade de tráfego (banda)
 - ▶ Número de contas de e-mail
 - ▶ Espaço para armazenamento do site, etc...
- ▶ Algumas empresas de hospedagem oferecem também serviço de registro e administração de domínios

Publicando o seu site na Web

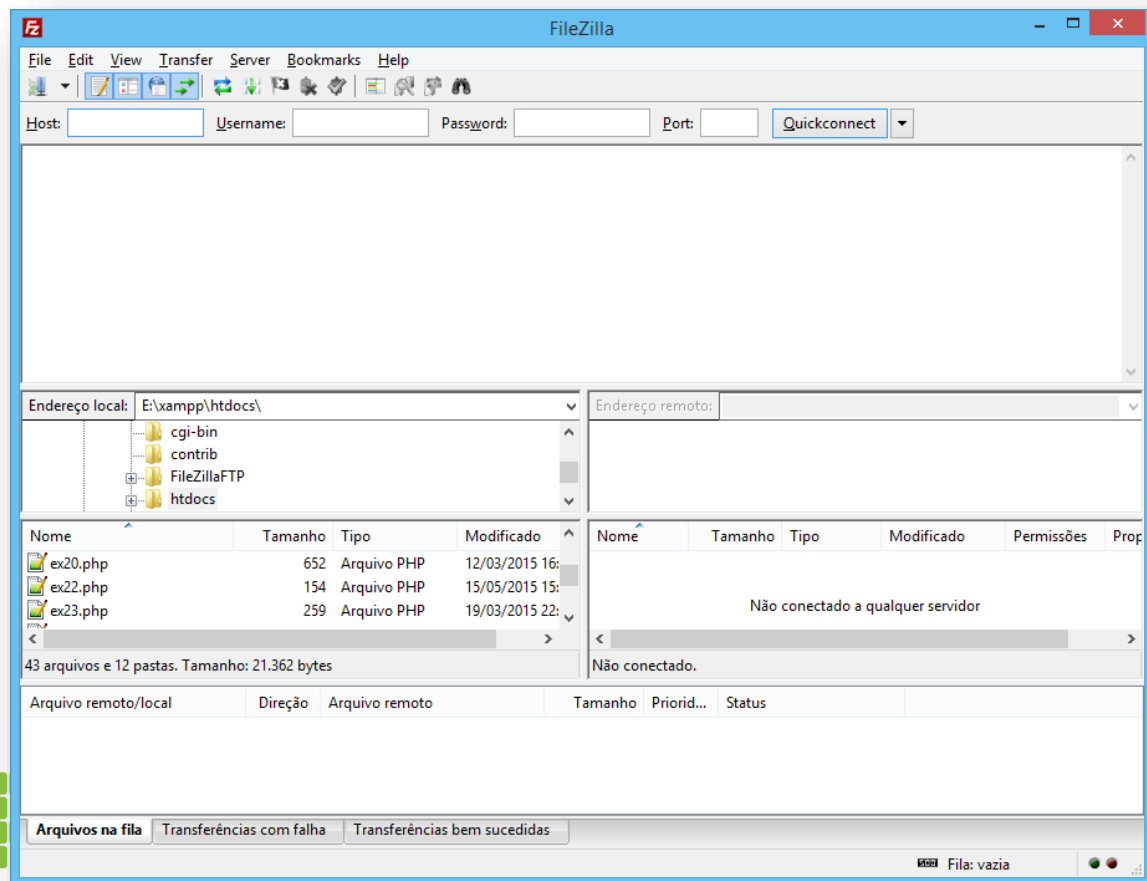
Serviços de hospedagem

- ▶ Alguns serviços de hospedagem pagos:
 - ▶ [Locaweb](#)
 - ▶ [UOL Host](#)
 - ▶ [Bluehost](#)
 - ▶ [DreamHost](#)
 - ▶ [HostGator](#)
- ▶ Alguns serviços de hospedagem gratuitos com suporte a PHP:
 - ▶ [000webhost](#)
 - ▶ [Hospedagem Livre](#)
 - ▶ Mais hospedagens gratuitas:
<http://www.hospedagensgratis.com.br/hospedagem-php-gratis/>

Publicando o seu site na Web

Transferindo arquivos com Cliente de FTP

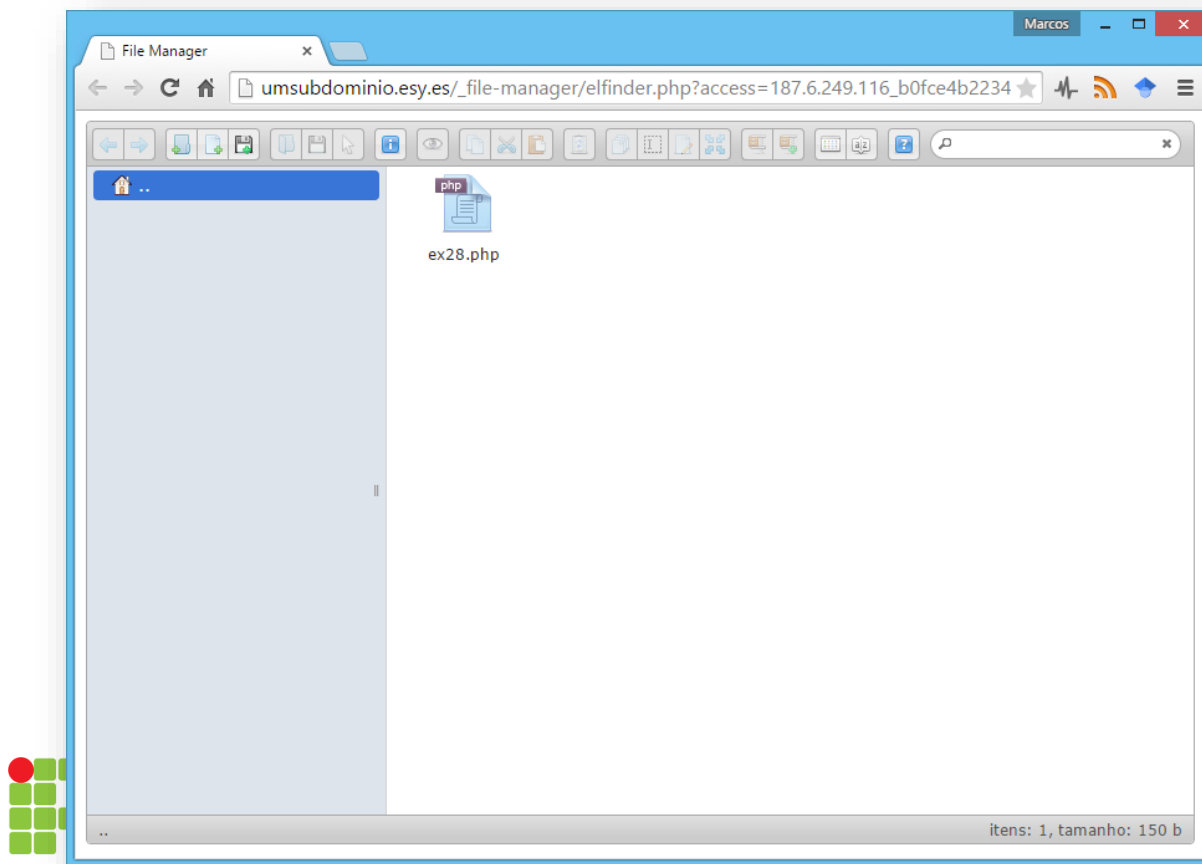
- ▶ Para transferir os arquivos do seu computador para o servidor de hospedagem contratado, é necessário utilizar um programa cliente de FTP
- ▶ Um bom (e gratuito) cliente de FTP é o FileZilla: <https://filezilla-project.org/>



Publicando o seu site na Web

Transferindo arquivos pela interface web do servidor de hospedagem

- ▶ Outra forma é utilizar o próprio painel de controle do servidor de hospedagem para transferir os arquivos



Enviando e-mails usando PHP

- mail

Envia um e-mail utilizando o servidor de e-mails local (de onde o *script* está executando)

Retorna `TRUE` se o e-mail foi aceito com sucesso para entrega, `FALSE` caso contrário

- ▶ Caso não possua um servidor de e-mails local, o e-mail simplesmente não será enviado, apesar da função retornar valor `TRUE` :-)
- ▶ Exemplo de envio de e-mail

```
<?php
$destinatario = 'email_do_destinatario@qualquercoisa.com.br';
$assunto = 'Teste de envio de e-mail com PHP';
$mensagem = 'Esse é meu primeiro e-mail enviado com PHP :-)';

mail($destinatario, $assunto, $mensagem);
?>
```

Enviando e-mails usando PHP

- ▶ Exemplo de envio de e-mail com cabeçalhos adicionais, para enviar um e-mail com conteúdo HTML

Para ter mais flexibilidade no envio de e-mails, utilize a classe [PHPMailer](#)

```
<?php
$destinatario = 'email_do_destinatario@qualquercoisa.com.br';
$assunto = 'Teste de envio de e-mail com PHP';
$mensagem = 'Esse é meu segundo e-mail enviado com PHP :-)';
$mensagem .= '<br>Agora posso usar <b><u>HTML</u></b>!';

//para enviar um e-mail HTML, devemos adicionar estes cabeçalhos
$headers = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";

//cabeçalhos adicionais
$headers .= 'From: Seu Nome <seunome@seusite.com.br>' . "\r\n";
$headers .= 'Cc: comcopiapara@qualquersite.com.br' . "\r\n";
$headers .= 'Bcc: copiaocultapara@outrosite.com.br' . "\r\n";

//envia o e-mail
mail($destinatario, $assunto, $mensagem, $headers);
?>
```

Observe o uso destes cabeçalhos para o envio de e-mails com conteúdo HTML.

Exercício 33

33) Faça um formulário de contato com os seguintes campos:

- Nome
 - Assunto
 - Mensagem
- ▶ O campo "Mensagem" deve ser de múltiplas linhas (`textarea`).
- ▶ Os dados digitados no formulário devem ser submetidos para outro arquivo PHP e este fará o envio dos dados para o seu endereço de e-mail, mas antes deve haver uma validação simples, pois todos os campos são obrigatórios.
- ▶ Crie uma conta em um dos serviços de hospedagem gratuitos apresentados, transfira os arquivos do formulário e o arquivo PHP que o processa e faça alguns testes.
- ▶ Observação: o e-mail pode levar alguns minutos para chegar até a sua caixa de entrada.

Integrando PHP com banco de dados

- ▶ Um sistema dinâmico possui entre suas principais responsabilidades a integração com algum banco de dados, para armazenar, por exemplo:
 - ▶ Usuários e senhas, dados de produtos, informações cadastrais de clientes, funcionários, fornecedores, vendas, etc...
- ▶ A diferenciação dos seguintes termos é essencial e se faz necessária:

- ▶ **Banco de Dados**

"É uma coleção de dados logicamente coerente que possui um significado implícito cuja interpretação é dada por uma determinada aplicação"

No modelo relacional, os bancos de dados são compostos por tabelas e estas, por sua vez, armazenam os dados

- ▶ **SGBD (Sistema de Gerenciamento (ou Gerenciador) de Banco de Dados)**

"Software construído para facilitar as atividades de definição, construção e manipulação de bancos de dados"

Exemplos: MySQL, PostgreSQL, Oracle, SQL-Server, etc...

Integrando PHP com banco de dados

- ▶ Estudaremos a integração do PHP com bancos de dados do tipo MySQL
- ▶ O MySQL já é parte integrante do pacote XAMPP, mas também pode ser obtido gratuitamente em seu site:
 - ▶ <https://www.mysql.com/>
- ▶ O XAMPP também inclui o phpMyAdmin, que é um *front-end* para o MySQL. Ele pode ser obtido gratuitamente em:
 - ▶ <http://www.phpmyadmin.net/>
 - ▶ Uma vez instalado e configurado, o phpMyAdmin pode ser acessado pelo endereço:
 - ▶ <http://localhost/phpmyadmin>
- ▶ Existem ainda outros *front-ends* para o MySQL, por exemplo:
 - ▶ [MySQL-Front](#)
 - ▶ [HeidiSQL](#)
- ▶ Outro *software* de destaque para administração de bancos de dados MySQL é o [MySQL Workbench](#), que é muito mais que um simples *front-end*



Integrando PHP com banco de dados

- ▶ A manipulação de dados armazenados em um banco de dados utilizando PHP exige algumas etapas:
 1. Efetuar a conexão com o servidor e selecionar o banco de dados
 2. Executar o comando SQL para
 - ▶ Consultar os registros
 - ▶ Inserir novos registros
 - ▶ Alterar um registro existente
 - ▶ Excluir um ou mais registros
 3. Obter os resultados
 4. Visualizar os resultados
 5. Encerrar a Conexão

Integrando PHP com banco de dados

1. Efetuar a conexão com o servidor e selecionar o banco de dados

- ▶ Primeiramente, é necessário criar uma variável que conterà a conexão com o SGBD MySQL
- ▶ Utiliza-se a função `mysqli_connect`
 - ▶ Objetivo: abre uma conexão com um servidor MySQL
 - ▶ Retorno: um identificador de conexão MySQL em caso de sucesso na conexão, ou `FALSE` em caso de falha.

Sintaxe

```
$nome_variavel = mysqli_connect("end_servidor", "usuario", "senha",  
"banco_de_dados");
```

Exemplo

```
$conexao = mysqli_connect("localhost", "root", "admin",  
"meu_banco");
```


Integrando PHP com banco de dados

2. Executar o comando SQL

- ▶ Uma vez conectado ao banco de dados e definido sobre qual banco as instruções serão realizadas, é necessário montar uma instrução (*query*) SQL
- ▶ As instruções SQL são *strings* que podem ser guardadas em variáveis, para melhor organização do código

Sintaxe

```
$nome_variavel = "instrucao sql";
```

Exemplo

```
$sql = "SELECT * FROM clientes ORDER BY nome";
```

Integrando PHP com banco de dados

3. Obter os resultados

- ▶ Para obter os resultados de uma instrução SQL, devemos aplicá-la ao banco de dados e armazenar a resposta em uma variável
- ▶ Utiliza-se a função mysqli_query
 - ▶ Objetivo: envia uma consulta MySQL
 - ▶ Retorno:

Variável especial, que mantém uma referência a um recurso externo.

- ▶ Para comandos SELECT, SHOW, DESCRIBE ou EXPLAIN, retorna um conjunto de resultados do tipo mysqli_result em caso de sucesso, ou FALSE em caso de falha;
- ▶ Para outros tipos de *query* SQL, como UPDATE, DELETE, DROP, etc., retorna TRUE em caso de sucesso ou FALSE em caso de erro.

Sintaxe

```
$nome_variavel = mysqli_query($variavel_conexao, $variavel_query_sql);
```

Exemplo

```
$rs = mysqli_query($conexao, $sql);
```

Integrando PHP com banco de dados

Para *queries* SQL como UPDATE, DELETE, DROP, etc., este passo é desnecessário.

4. Visualizar os resultados

- ▶ Para exibir o resultado de uma consulta SQL (`SELECT`), é necessário separar os registros por linha
- ▶ Utiliza-se a função `mysqli_fetch_assoc`
 - ▶ Objetivo: obtém um linha do resultado como uma matriz associativa
 - ▶ Retorno: uma matriz associativa de *strings* que corresponde à linha obtida, ou `NULL` se não houverem mais linhas.

Sintaxe

```
$nome_variavel = mysqli_fetch_assoc($variavel_resultado_query_sql);
```

Exemplo

```
while($reg = mysqli_fetch_assoc($rs)){  
    echo $reg['nome'] . '<br>' . $reg['cpf'];  
}
```

Integrando PHP com banco de dados

5. Encerrar a conexão

- ▶ Ao terminar as operações em uma tabela, convém fechá-la, assim como a conexão utilizada
- ▶ Utilizam-se as funções:
 - ▶ mysqli_free_result
 - ▶ Objetivo: Libera um resultado da memória
 - ▶ Retorno: *não há valor retornado*
 - ▶ mysqli_close
 - ▶ Objetivo: Fecha a conexão MySQL
 - ▶ Retorno: TRUE em caso de sucesso ou FALSE em caso de falha

Sintaxe

```
mysqli_free_result($variavel_resultado_query_sql);  
mysqli_close($variavel_conexao);
```

Exemplo

```
mysqli_free_result($rs);  
mysqli_close($conexao);
```

Integrando PHP com banco de dados

Resumo das Operações

1	Efetuar a conexão com o servidor MySQL e selecionar o banco de dados <code>\$nome_variavel = mysqli_connect("end_servidor", "usuario", "senha", "banco_de_dados");</code> Exemplo: <code>\$conexao = mysqli_connect("localhost", "root", "admin", "meu_banco");</code>
2	Criar a instrução SQL <code>\$nome_variavel = "instrucao sql";</code> Exemplo: <code>\$sql = "SELECT * FROM clientes ORDER BY nome";</code>
3	Guardar o resultado obtido em uma variável <code>\$nome_variavel = mysqli_query(\$variavel_conexao, \$variavel_query_sql);</code> Exemplo: <code>\$rs = mysqli_query(\$conexao, \$sql);</code>
4	Preparar os dados para serem exibidos <code>\$nome_variavel = mysqli_fetch_assoc(\$variavel_resultado_query_sql);</code> Exemplo: <pre>while(\$reg = mysqli_fetch_assoc(\$rs)){ echo \$reg['nome'] . '
' . \$reg['cpf']; }</pre>
5	Encerrar a conexão com o banco <code>mysqli_free_result(\$variavel_resultado_query_sql);</code> <code>mysqli_close(\$variavel_conexao);</code> Exemplo: <code>mysqli_free_result(\$rs);</code> <code>mysqli_close(\$conexao);</code>

IMPORTANTE:

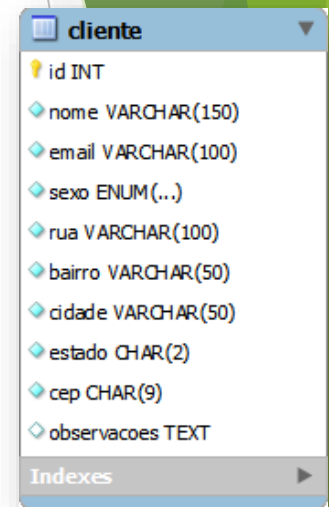
Insira pelo menos 10 registros na tabela, utilizando o formulário. Isso servirá para testar seu código e também para popular a tabela.

Exercício 34

- 34) Obtenha os arquivos `meu_formulario.php` e `exibe_dados.php`, utilizados no exercício 32. Modifique o arquivo `exibe_dados.php` para, **após validar e mostrar os dados submetidos pelo formulário** (isso já deve estar pronto), gravá-los no banco de dados, exibindo uma mensagem de sucesso ou falha no processo de gravação.

O SQL para criação do banco de dados `meubanco` e da tabela `cliente` é o seguinte:

```
CREATE DATABASE IF NOT EXISTS `meubanco`;
CREATE TABLE IF NOT EXISTS `meubanco`.`cliente` (
  `id` INT NOT NULL AUTO INCREMENT,
  `nome` VARCHAR(150) NOT NULL,
  `email` VARCHAR(100) NOT NULL,
  `sexo` ENUM('masculino','feminino') NOT NULL,
  `rua` VARCHAR(100) NOT NULL,
  `bairro` VARCHAR(50) NOT NULL,
  `cidade` VARCHAR(50) NOT NULL,
  `estado` CHAR(2) NOT NULL,
  `cep` CHAR(9) NOT NULL,
  `observacoes` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
```



O SQL para inserção de um registro na tabela `cliente` é o seguinte:

```
INSERT INTO `meubanco`.`cliente`
  (`nome`, `email`, `sexo`, `rua`, `bairro`, `cidade`, `estado`,
  `cep`, `observacoes`)
VALUES
  ('Marcos', 'marcos.vieira@ifgoiano.edu.br', 'masculino', 'Av.
  Pará', 'Centro', 'Iporá', 'GO', '76200-000', 'Sem observações no
  momento');
```

Exercício 35 e 36

- 35) Crie o arquivo `lista_clientes.php`, e mostre os dados de todos os clientes gravados na tabela `clientes`.

O SQL para esta consulta é o seguinte:

```
SELECT * FROM `meubanco`.`cliente` ORDER BY `nome`;
```

Exemplo:

Id: 1
Nome: Marcos
E-mail: marcos.vieira@ifgoiano.edu.br
Sexo: masculino
Rua: Av. Pará
Bairro: Centro
Cidade: Iporá
Estado: GO
CEP: 76200-000
Observações: Sem observações

Id: 2 (...)

DICA: Reproveite o código utilizado para exibir os dados do formulário, contido no arquivo `exibe_dados.php`, do exercício 34.

- 36) Modifique o arquivo `lista_clientes.php` para mostrar os dados de todos os clientes gravados na tabela `clientes` em forma de uma tabela, onde cada linha representa os dados de um registro da tabela.

ID	Nome	E-mail	Sexo	Rua	Bairro	Cidade	Estado	CEP	Observações
1	Marcos	marcos.vieira@ifgoiano.edu.br	masculino	Av. Pará	Centro	Iporá	GO	76200-000	Sem observações
2	(...)								

DICA: Faça o código HTML da tabela estática. Depois, coloque a linha do registro (`<tr>` até `</tr>`) dentro do `while` do `mysql_fetch_assoc`, fazendo então as alterações necessárias para exibição dos dados.

Referências bibliográficas

- ▶ DALL'OGGIO, Pablo. **PHP Programando com Orientação a Objetos: Inclui Design Patterns**. 2ª Ed. São Paulo: Novatec, 2009.
- ▶ SOARES, Wallace. **PHP 5: conceitos, programação e integração com banco de dados**. 7ª Ed. São Paulo: Érica, 2013.
- ▶ OLIVIERO, Carlos A. J. **Faça um Site: PHP 5.2 com MySQL 5.0: Comércio Eletrônico: Orientado por Projeto**. São Paulo: Érica, 2010.
- ▶ SILVA, Mauricio S. **Criando sites com HTML: sites de alta qualidade com HTML e CSS**. São Paulo: Novatec, 2008.
- ▶ _____. **Construindo sites com CSS e (X)HTML: sites controlados por folhas de estilo em cascata**. São Paulo: Novatec, 2008.
- ▶ PHP. **Manual do PHP**. Disponível em <<http://www.php.net>>.