



UFRR

Universidade Federal de Roraima
Pró-Reitoria de Ensino e Graduação
Núcleo de Educação a Distância
Centro de Ciências e Tecnologia
Departamento de Ciência da Computação

PROGRAMAÇÃO PARA WEB

Curso de Licenciatura em
Informática a Distância

Flávio Izo

PROF. FLÁVIO IZO

LICENCIATURA EM INFORMÁTICA
Programação Web

CACHOEIRO DE ITAPEMIRIM
IFES
2012

Governo Federal
Ministro da Educação
Aloizio Mercadante

Instituto Federal do Espírito Santo - Ifes

Reitor
Denio Rebello Arantes

Pró-Reitora de Ensino
Cristiane Tenan Schlittler dos Santos

Diretor-geral do Campus Cachoeiro de Itapemirim
Mário Jorge de Moura Zuany

Diretora do Cead – Centro de Educação a Distância
Yvina Pavan Baldo

Coordenadores da UAB – Universidade Aberta do Brasil
Marize Lyra Silva Passos
José Mario Costa Junior

Curso de Licenciatura em Informática

Coordenação de Curso
Alexandre Fraga de Araujo

Designer Instrucional
Marcelo Albuquerque Schuster

Professor Formador/Autor
Flávio Izo

I98p Izo, Flávio.
2012 Programação web. / Flávio Izo. – Cachoeiro de Itapemirim:
 Ifes campus Cachoeiro de Itapemirim, 2012.
 120 p. il.

Inclui bibliografia.

ISBN: 978-85-62934-61-2

1. Programação web. 2. Programação – HTML. 3.

Programação – Java. 4. Programação – PHP. I. Izo, Flávio. II.

Instituto Federal do Espírito Santo. III. Título.

CDD 005.1

DIREITOS RESERVADOS

Instituto Federal do Espírito Santo – Ifes

Av. Rio Branco, nº 50 - Santa Lúcia - Vitória - ES - CEP. 29056-255 – Vitória – ES - Telefone: (27) 3227-5564

Créditos de autoria da editoração

Capa: Juliana Cristina da Silva

Projeto gráfico: Juliana Cristina e Nelson Torres

Iconografia: Nelson Torres

Editoração eletrônica: Cead/Ifes

Revisão de texto:

Esther Ortlieb Faria de Almeida

COPYRIGHT – É proibida a reprodução, mesmo que parcial, por qualquer meio, sem autorização escrita dos autores e do detentor dos direitos autorais.

Olá, Aluno(a)!

É um prazer tê-lo(a) conosco.

O Ifes – Instituto Federal do Espírito Santo – oferece a você, em parceria com as Prefeituras e com o Governo Federal, o Curso de Licenciatura em Informática, na modalidade a distância. Apesar de este curso ser ofertado a distância, esperamos que haja proximidade entre nós, pois, hoje, graças aos recursos da tecnologia da informação (e-mails, chat, web conferência, etc.) podemos manter uma comunicação efetiva.

É importante que você conheça toda a equipe envolvida neste curso: coordenadores, professores formadores, tutores, porque, quando precisar de algum tipo de ajuda, saberá a quem recorrer.

Na EaD – Educação a Distância, você é o grande responsável pelo sucesso da aprendizagem. Por isso, é necessário que se organize para os estudos e para a realização de todas as atividades, nos prazos estabelecidos, conforme orientação dos Professores Formadores e Tutores.

Fique atento às orientações de estudo que se encontram no Manual do Aluno!

A EaD, pela sua característica de amplitude e pelo uso de tecnologias modernas, representa uma nova forma de aprender, respeitando, sempre, o seu tempo.

Desejamos-lhe sucesso e dedicação!

Equipe do Ifes

ICONOGRAFIA

Veja, abaixo, alguns símbolos utilizados neste material para guiá-lo em seus estudos.

Fala Professor



Fala do Professor.

Conceitos



Conceitos importantes. Fique atento!

Atividades



Atividades que devem ser elaboradas por você, após a leitura dos textos.

Indicações



Indicação de leituras complementares, referentes ao conteúdo estudado.

Atenção



Destaque de algo importante, referente ao conteúdo apresentado. Atenção!

Reflexão



Reflexão/questionamento sobre algo importante, referente ao conteúdo apresentado.

Anotações



Espaço reservado para as anotações que você julgar necessárias.

APRESENTAÇÃO

Olá!

Meu nome é Flávio Izo, responsável pela disciplina Programação Web. Atuo como professor do IFES há dois anos e já lecionei em outra instituição de ensino superior (Centro Universitário São Camilo-ES). Sou graduado em Sistemas de Informação (2005) e Pós Graduado em Docência do Ensino Superior (2007), ambos pelo Centro Universitário São Camilo-ES. Minhas áreas de interesse são: Lógica de Programação, Modelagem de Dados, Banco de Dados e Programação Web.

Nesta disciplina, você aprenderá técnicas para o desenvolvimento de projetos Web, conhecendo os principais conceitos e sua aplicabilidade.

A programação Web está dividida em duas grandes áreas: a linguagem Client-Side e a linguagem Server-Side. A primeira segue as premissas de programação que funcionarão do lado do cliente (usuário); a segunda se refere à programação que está relacionada ao servidor Web (onde os arquivos ficam armazenados).

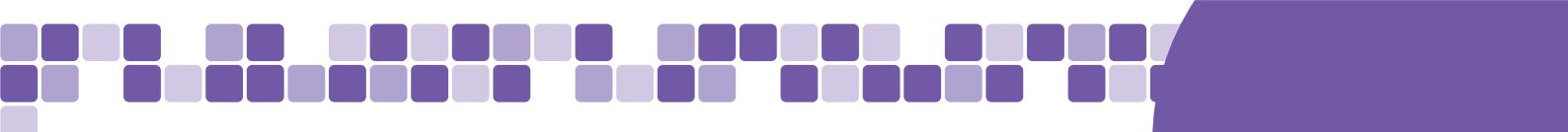
Dessa forma, o objetivo deste material é analisar e fazer a integração dessas duas áreas, formando uma programação Web que contemple os objetivos do usuário final.

Também serão abordadas outras técnicas complementares à nossa disciplina, tais como: conexão com banco de dados, utilização de sessões (permite ficar conectado por um intervalo de tempo, sem precisar efetuar um novo login); upload de arquivos; e uma introdução à Orientação a Objeto.

Assim, desejo-lhe bastante sucesso!

Prof. Flávio Izo

Bons estudos!



Sumário

Cap. 1 - INTRODUÇÃO 9

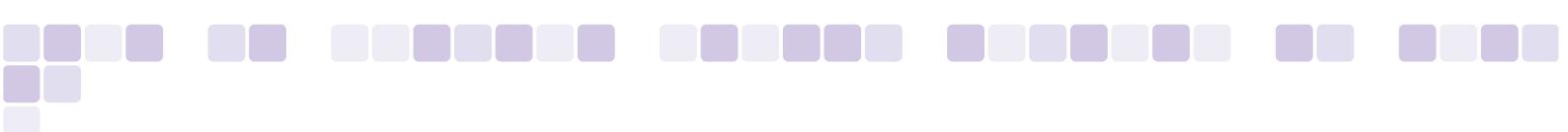
- 1.1 O QUE É PROGRAMAÇÃO WEB? 9
- 1.2 O QUE É LINGUAGEM *CLIENT-SIDE*? 10
- 1.3 O QUE É LINGUAGEM *SERVER-SIDE*? 11
- 1.4 QUAL A IMPORTÂNCIA DA PROGRAMAÇÃO WEB? 13
- 1.5 A ESTRUTURA DA URL 14

Cap. 2 - LINGUAGEM *CLIENT-SIDE* 17

- 2.1 ESTRUTURA BÁSICA DO HTML 17
- 2.2 DIFERENÇA ENTRE JAVASCRIPT E JAVA 23
- 2.3 INICIANDO COM O JAVASCRIPT 24
- 2.4 DOM (*DOCUMENT OBJECT MODEL*) 25
- 2.5 INSERINDO UM TEXTO NA PÁGINA WEB 29
- 2.6 FORMATAÇÃO DOS CARACTERES 29
- 2.7 FORMATAÇÃO DO DOCUMENT 30
- 2.8 VARIÁVEIS 31
- 2.9 NOMES RESERVADOS (PALAVRAS RESERVADAS) 32
- 2.10 OPERADORES 33
- 2.11 FUNÇÕES 34
- 2.12 EVENTOS 36
- 2.13 ESTRUTURA CONDICIONAL 37
- 2.14 FORMULÁRIOS 38
- 2.15 ERROS COMUNS COMETIDOS PELOS
PROGRAMADORES 47
- 2.16 CAIXAS DE MENSAGEM 48

Cap. 3 - LINGUAGEM *SERVER-SIDE* 51

- 3.1 INSTRUÇÕES INICIAIS PARA UTILIZAR O PHP 51
- 3.2 CONFIGURANDO O PHP 54
- 3.3 UTILIZAÇÃO DE FERRAMENTAS PARA DESENVOLVIMENTO
WEB 57
- 3.4 INTRODUÇÃO À SINTAXE BÁSICA 58
 - 3.4.1 Imprimindo conteúdo na tela: 59
- 3.5 TIPOS DE DADOS 60
- 3.6 VARIÁVEIS 61
- 3.7 CONSTANTES 65



| | |
|---|-----------|
| 3.8 CONCATENANDO VALORES | 67 |
| 3.9 ARRAYS SUPER GLOBAIS | 67 |
| 3.10 OPERADORES | 68 |
| 3.10.1 Operadores Aritméticos av: | 68 |
| 3.11 ESTRUTURAS CONDICIONAIS | 70 |
| 3.12 ESTRUTURAS DE REPETIÇÃO | 71 |
| 3.13 FUNÇÕES | 73 |
| 3.14 INTERNACIONALIZAÇÃO DE FORMULÁRIOS HTML | 78 |
| 3.15 PASSANDO INFORMAÇÕES ENTRE AS PÁGINAS | 79 |
| 3.15.1 Método GET | 80 |
| 3.15.2 Método POST | 81 |
| 3.16 SESSÕES | 83 |
| 3.17 UPLOAD DE ARQUIVOS | 85 |
| 3.18 PROBLEMAS COMUNS COM PHP | 88 |

Cap. 4 - UTILIZANDO O BANCO DE DADOS **91**

| | |
|--|------------|
| 4.1 FERRAMENTAS PARA ADMINISTRAR O BANCO DE DADOS | 91 |
| 4.2 HABILITANDO UMA CONEXÃO | 92 |
| 4.3 COMUNICAÇÃO DA LINGUAGEM COM O BANCO DE DADOS | 93 |
| 4.3.1 Conectando ao banco de dados MySql | 93 |
| 4.3.2 Selecionando um banco de dados | 94 |
| 4.3.3 Executando uma query | 94 |
| 4.3.4 Organizando uma consulta | 95 |
| 4.3.5 Fechando uma conexão | 100 |
| 4.3.6 Verificando erros | 100 |
| 4.3.7 Outras funções do MySql | 101 |

Cap. 5 - IMPLEMENTAÇÃO DE ESTUDO DE CASO **103**

Cap. 6 - CLASSES E OBJETOS **115**

REFERÊNCIAS **119**





INTRODUÇÃO

Olá!

Neste capítulo inicial, vamos compreender alguns conceitos referentes à programação WEB e suas aplicabilidades.

Assim, nosso objetivo neste capítulo é apresentar conceitos que fundamentam a programação WEB e discutir sobre eles.

Começaremos falando, de modo geral, sobre o que vem a ser programação WEB. Posteriormente, abordaremos também outras áreas que envolvem essa programação, tais como linguagem Client-Side e da linguagem Server-Side.

Para finalizar o capítulo, discutiremos sobre a importância que a programação Web tem nos dias atuais.

Os conhecimentos adquiridos neste capítulo serão úteis para a aprendizagem dos demais conteúdos trabalhados nesta disciplina.

Bons estudos!

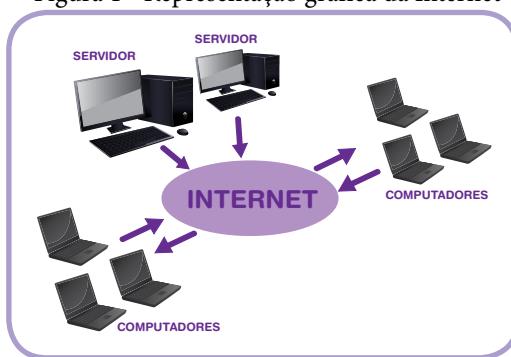
Fala Professor

*Angue risus ac
e velit at tellus.
massa portitor
sectetur magna.*

1.1 O QUE É PROGRAMAÇÃO WEB?

Antes de discutir acerca da programação Web, necessitamos citar o conceito Internet. A Internet é um imenso sistema de redes de computadores que ficam permanentemente interligados a nível mundial e que funcionam como emissores e receptores de informações. A Internet permite interligar sistemas de informação de todo o mundo, possibilitando a comunicação e a troca de informações a qualquer momento, conforme figura 1.

Figura 1 - Representação gráfica da Internet



Sendo assim, a Internet possibilita que diversos serviços sejam executados, dentre os quais poderíamos citar: Web, mensagens instantâneas, acesso remoto, transferência de arquivos, correio eletrônico, etc. Vamos, aqui, enfatizar a Web, haja vista que é o nosso foco nesta disciplina.

A Web, que por sua vez também é chamada de WWW (*World Wide Web*, ou, em português, ‘Rede de abrangência mundial’) é uma rede interligada por computadores que fornece informações diversas através de hipertexto.

Durante esta disciplina, desenvolveremos alguns exemplos de scripts e o nosso objetivo final será demonstrar um projeto mais robusto, com características personalizáveis. Mas, o que site tem a ver com a Web? Simples: um site é um conjunto de hipertextos que são acessados pelo protocolo HTTP (*HyperText Transfer Protocol*), e o conjunto de sites é que forma a grande Rede de abrangência Mundial (WWW). Por isso, chamamos os sites de websites.

Os websites podem ser divididos de duas formas: websites estáticos e websites dinâmicos. Os dinâmicos possuem personalização, permitindo ao usuário visualizar informações pessoais, interagir com o sistema, etc. Já os estáticos são mais simples, tendo como objetivo ser mais informativo, não permitindo personalização do usuário no sistema. Esse tipo de site é para ser utilizado da mesma forma pelo público geral, não importando se o usuário é homem ou mulher, se é a Maria ou se é o João ou, ainda, o Tiago. Ou seja, todos visualizarão o mesmo website.

Atenção



A maioria (para não dizer 100%) dos websites dinâmicos utilizam uma linguagem server-side para programar a interação (personalização) do usuário com o site.

No próximo item, abordaremos as particularidades entre as linguagens *client-side* e *server-side*.

1.2 O QUE É LINGUAGEM *CLIENT-SIDE*?

De maneira resumida, *client-side* é a programação Web que funciona do lado “cliente”. Podemos citar como exemplo de linguagens que se encaixam nessa situação: HTML, CSS e *Javascript*.

Mas e o PHP, ASP, JavaWeb? Essas são exemplos de linguagens *server-side*, ou seja, funcionam do lado servidor (falaremos mais sobre linguagem servidor no próximo item).

A linguagem *client-side* é executada no computador do próprio usuário, sendo assim, ela é muita usada em situações nas quais a linguagem *server-side* não tem alcance. Outra vantagem da linguagem *client-side* é a rapidez e agilidade na execução dos scripts, pois não necessita do uso de uma boa Internet para funcionar, já que roda no próprio browser do usuário.

No *javascript*, por ser *client side*, não é diferente, pois todo o código está visível no próprio navegador e o usuário pode ver e manipular o código também. Isso faz com que as linguagens *client-side* sejam inseguras. Por isso necessitamos utilizar as linguagens *client* e *server* para obter a combinação perfeita.

Browser

Também chamado de navegador. É o programa que utilizamos para ter acesso aos recursos da Internet, tais como acessar um website.

Exemplo: Mozilla Firefox, Internet Explorer, Google Chrome etc.



Conceitos

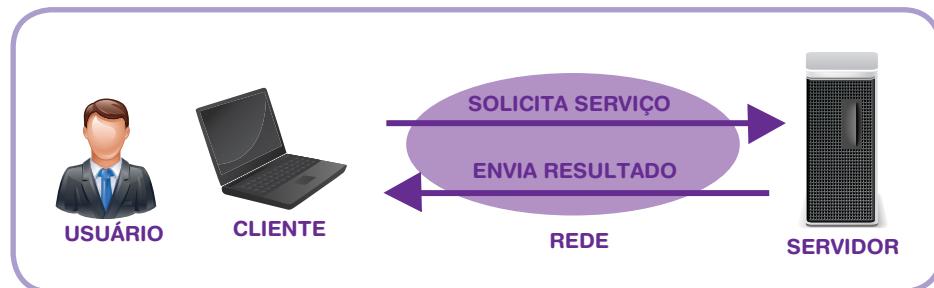
1.3 O QUE É LINGUAGEM SERVER- SIDE?

Ao contrário da *client-side*, a *server-side* utiliza não somente a máquina do cliente (usuário), mas também o servidor Web. Esse servidor será o responsável pela execução de códigos (*scripts*) mais abrangentes, tais como cadastro de clientes, efetuar compras, etc. Isso demonstra que, se quisermos incluir em um site: armazenamento no banco de dados, itens como autenticação de usuários, restringir acesso de usuários a determinadas páginas, possibilitar compras entre outros; devemos possuir um servidor Web com suporte para a linguagem Server que desejamos utilizar.

Entre os exemplos de linguagens *server-side*, podemos citar: PHP, ASP, Java Web e C#.

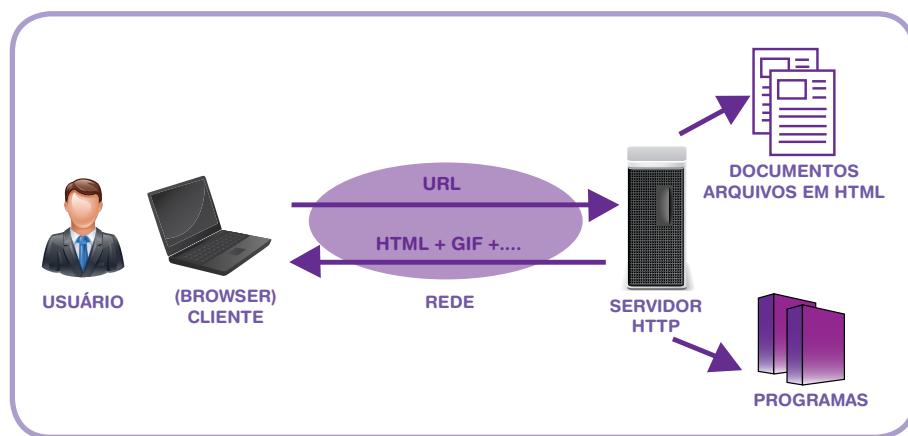
Na figura 2, exemplificamos como se dá a relação entre cliente/servidor.

Figura 2 - Representação cliente/servidor



Essa mesma solicitação de serviço pode ser exemplificada como uma requisição de uma url. Essa url é interpretada e identificada no servidor DNS, redirecionado para o servidor Web e devolvido como html.

Figura 3 - Requisição de uma página Web



Pense na maneira como você acessa o site do CEAD:

- Primeiro você digita o endereço <http://cead.ifes.edu.br/>
- Esse endereço é recebido e redirecionado pelo servidor de DNS para o local onde estão armazenados os arquivos do site (figuras, scripts, vídeos etc.).
- Por fim, tudo é convertido em html e a página do CEAD é devolvida para o usuário como código html.

Figura 4 - Página Web do CEAD



Figura 5 - Código fonte (html) da página do CEAD

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-br" lang="pt-br" >
<head>
<!--[Obrigat&nbsp;rio]--> Este include adiciona os marcadores html's definidos do sistemas<![endObrigat&nbsp;rio]-
<base href="http://cead.ifes.edu.br/" />
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta name="robots" content="index, follow" />
<meta name="keywords" content="cead, ifes, instituto federal do espirito santo, centro de educação a distância" />
<meta name="description" content="Portal Institucional do Centro de Educação a Distância do Ifes" />
<meta name="generator" content="Joomla! 1.5 - Open Source Content Management" />
<title>Cead :: Centro de Educação a Distância do Ifes</title>
<link href="/index.php?format=feed&type=rss" rel="alternate" type="application/rss+xml" title="RSS 2.0" />
<link href="/index.php?format=feed&type=atom" rel="alternate" type="application/atom+xml" title="Atom 1.0" />
<link href="/templates/portal/favicon.ico" rel="shortcut icon" type="image/x-icon" />
<link rel="stylesheet" href="/media/system/css/modal.css" type="text/css" />
<link rel="stylesheet" href="http://cead.ifes.edu.br/components/com_k2/css/k2.css" type="text/css" />
<script type="text/javascript" src="/media/system/js/mootools.js"></script>
<script type="text/javascript" src="/media/system/js/modal.js"></script>
<script type="text/javascript" src="http://cead.ifes.edu.br/components/com_k2/js/k2.js"></script>
<script type="text/javascript" src="/media/system/js/caption.js"></script>
<script type="text/javascript" src="http://cead.ifes.edu.br/plugins/system/iewarning/js/warning.js"></script>
<script type="text/javascript" src="http://cead.ifes.edu.br/components/com_jxtcprimetime/jxtcawfobject.js"></script>
<script type="text/javascript">
```

window.addEvent('domready', function() {
 SqueezeBox.initialize();
 \$\$('a.modal').each(function(el) {
 el.addEvent('click', function(e) {
 new Event(e).stop();
 SqueezeBox.fromElement(el);
 });
 });
});

Na figura 4 a página Web está aberta, como a vemos normalmente quando acessamos o site do CEAD. Já na figura 5, estamos vendo o código fonte, ou seja, o que foi obtido como resultado após a requisição de abertura do site.

Para visualizar o código fonte usando o Internet Explorer, basta acessar o menu “Página” e clicar em “Exibir código fonte”.

Para visualizar o código fonte usando o Mozilla Firefox, basta apertar as teclas “CTRL + U”.

Para visualizar o código fonte usando o Google Chrome, acesse o menu “Ferramentas” e clique em “Exibir código fonte”.



Atenção

Como podemos perceber neste tópico, o cliente faz a solicitação do serviço, assim o servidor processa a requisição e devolve o resultado através de código HTML (*HyperText Markup Language*, que significa Linguagem de Marcação de Hipertexto).

1.4 QUAL A IMPORTÂNCIA DA PROGRAMAÇÃO WEB?

Nos últimos anos, pôde-se perceber a expansão tecnológica ocorrida na Web. Vemos sites como o facebook, twitter, linkedln, google e outros tantos (chega de exemplos, afinal, poderíamos ficar citando várias páginas com nomes de sites promissores, haja vista que surgem novos

projetos web todos os dias) nascem com propostas (normalmente boas) e conquistam os usuários/internautas.

Quando falamos que normalmente os projetos são bons é porque, no atual mercado tecnológico, não há mais espaço para propostas ruins, que não mantenham o foco no usuário.

Poderíamos citar também como importância da Web o fato de conseguirmos achar informações com apenas alguns “cliques no mouse”, mesmo que essas informações estejam geograficamente distantes de nós.

Atualmente, as pessoas utilizam a Internet para executar tarefas que há alguns anos deveriam ser feitas de forma presencial. Podemos citar, por exemplo:

- recrutamento de pessoas (currículo online, análise de páginas pessoas na rede social etc.);
- serviços de banco (pagamentos, extratos, empréstimos etc.);
- cursos a nível técnico e superior (avanço da EaD – Educação a Distância);
- compra de produtos/serviços;
- conferências síncronas com áudio e vídeo; entre outros.

Haja vista essa nossa explanação, não restam dúvidas de que a Web existe e está presente no nosso dia a dia. Façamos, então, uma reflexão sobre como podemos desenvolver ferramentas online que auxiliem a comunidade mundial na execução de atividades que deveriam ser feitas de maneira presencial, ou mesmo desenvolver projetos para o puro lazer e/ou para troca de informações.

1.5 A ESTRUTURA DA URL

Na figura 4 falamos de url e citamos o exemplo do site do CEAD (<http://cead.ifes.edu.br/>). Agora vamos discutir acerca da estrutura que forma uma url. Para isso, iremos estabelecer uma url como exemplo:

<http://www.ifes.edu.br/mod4/pw/arq1.pdf>

Onde:

- **http:** => Protocolo;
- **//** => raiz do endereço da internet;
- **www.ifes.edu.br** => domínio ou IP
- **mod4/pw/arq1.pdf** => caminho a percorrer

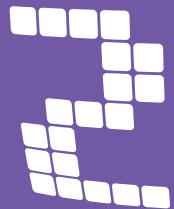
1. Explique a diferença entre websites dinâmicos e estáticos.
 2. Descreva, com suas palavras e fundamentado no que foi discutido neste capítulo 1, a importância da programação Web nos dias atuais.
 3. Como é formada a estrutura da url. Dê 3 exemplos que contemplam todas as partes da url.
 4. A partir do que foi discutido aqui neste capítulo, elabore um parágrafo explicando como funciona a relação cliente-servidor.
 5. Um desenvolvedor criou um *script*, utilizando a linguagem *javascript*, para gravar dados dos clientes em um banco de dados. Essa afirmação está correta? Justifique.



Atividades



Anotações



LINGUAGEM *CLIENT-SIDE*

Olá!

Neste capítulo dois, vamos compreender como criar um website utilizando a linguagem client-side.

Nesta abordagem, entenderemos primeiramente como é a estrutura básica do HTML (base para qualquer site). Logo em seguida, discorreremos acerca do javascript (linguagem client-side) e suas particularidades, tais como formatação, variáveis, operadores, eventos, condições, estruturas de repetição e formulários.

Os conteúdos estudados de javascript são essenciais para a criação de qualquer projeto Web.

Os conhecimentos adquiridos neste capítulo serão úteis para a aprendizagem dos demais conteúdos trabalhados nesta disciplina.

Bons estudos!

Angue risus at
e velit at tellus.
massa porttitor
isectetur magna.

Fala Professor

No capítulo anterior citamos a linguagem *client-side*. Agora demonstraremos como criar um website utilizando esse tipo de linguagem.

Para começar, necessitamos conhecer a estrutura básica do HTML.

2.1 ESTRUTURA BÁSICA DO HTML

Para desenvolver nossas páginas HTML podemos utilizar um editor de textos qualquer. Inicialmente, eu irei utilizar o *notepad* (Bloco de Notas).

Existem programas profissionais mais robustos que possibilitam criar sites de maneira mais fácil e rápida. Esses programas, normalmente, necessitam da compra de licença para serem utilizados. Vamos citar alguns editores:

Macromedia Dreamweaver; NVU; Microsoft FrontPage.



Atenção

Os editores permitem criar e editar o código das páginas do seu site. A maioria dos programas mais sofisticados permite trabalhar com o conceito WYSIWYG que é o acrônimo da expressão em inglês “*What You See Is What You Get*”, ou seja, “o que você vê é o que você obtém”. Esse conceito é a garantia de que o que você está criando é o que você terá ao publicar o site.

Os editores visuais são mais fáceis para criar um website, e recomendados para facilitar o desenvolvimento de projetos que requerem rapidez. Em contrapartida, mesmo utilizando esses editores visuais, é necessário conhecer a fundo sobre a linguagem, pois, em determinadas situações nas quais o editor não tiver o recurso que você gostaria de criar, você terá que utilizar o código fonte para criar manualmente a programação do recurso desejado.

Mas, voltando à estrutura básica do HTML!

Tags são rótulos usados para informar ao navegador como deve ser apresentado o website. Todas as *tags* têm o mesmo formato: começam com um sinal de menor “<” e acabam com um sinal de maior “>”.

Genericamente falando, existem dois tipos de *tags*:

- *tags* de abertura: <comando>
- *tags* de fechamento: </comando>

A diferença entre elas é que na *tag* de fechamento existe uma barra “/”.

Todo documento (página Web) começa com as *tag* <HTML> e termina com a *tag* </HTML>.

A seguir vemos a estrutura básica do HTML.

| | |
|--------------------|---|
| <html> <head> | A tag <html> indica o início/fim do conteúdo do site; |
| </head> <body> | A tag <head> indica o início/fim do cabeçalho do site. Fornece informações sobre o site. Ex.: Onde fica o título da página; |
| </body> </html> | A tag <body> indica o início/fim do corpo do site. Todo o conteúdo do site fica entre essas tags. |

Tudo que estiver contido entre uma *tag* de abertura e uma *tag* de fechamento será processado segundo o comando contido na *tag*.

Existe uma exceção: para algumas *tags*, a abertura e o fechamento se dá na mesma *tag*. Tais *tags* contêm comandos que não necessitam de um conteúdo para serem processados, isto é, são *tags* de comandos isolados, por exemplo, um pulo de linha é conseguido com a *tag*
 ou para uma régua utilizamos a *tag* <hr />.

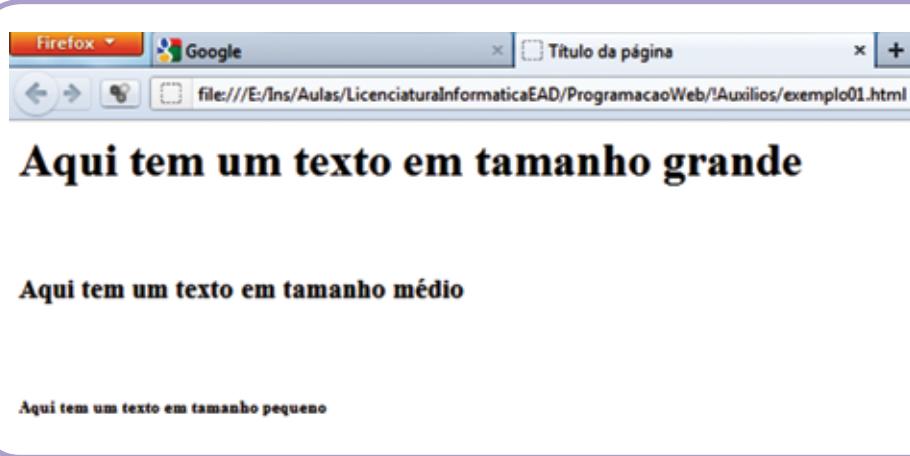
Figura 6 - Código fonte (html)

```

1  <html xmlns="http://www.w3.org/1999/xhtml">
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4      <title>Titulo da página</title>
5    </head>
6
7    <body>
8
9      <h1>Aqui tem um texto em tamanho grande</h1><br />
10     <h3>Aqui tem um texto em tamanho médio</h3><br />
11     <h6>Aqui tem um texto em tamanho pequeno</h6>
12
13   </body>
14 </html>

```

Figura 7 - Página da figura 6 aberta no navegador



Podemos aperfeiçoar nossa página colocando a *tag* <p> que gera um parágrafo.

Figura 8 - Exemplo de página (código html)

```

1  <html xmlns="http://www.w3.org/1999/xhtml">
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4      <title>Titulo da página</title>
5    </head>
6
7    <body>
8
9      <h1>Um cabeçalho</h1>
10     <p>texto, texto texto, texto</p>
11     <h2>Subtítulo</h2>
12     <p>texto2, texto2 texto2, texto2</p>
13
14   </body>
15 </html>

```

E o resultado da figura 8:

Figura 9 - Página da figura 8 aberta no navegador



Às vezes se torna necessário fazer comentários no código. Os comentários são feitos da seguinte forma:

```
<!--  
Meus comentários  
-->
```

Para complementar nossa explicação, vamos dar alguns outros exemplos de *tags* que podem ser utilizadas para criar uma página web.

| | |
|---|--|
| <pre><i>texto itálico</i> texto em negrito <i>texto itálico e negrito </i> -> quebra de linha <hr /> -> insere uma régua horizontal</pre> | |
| <pre> Um item de lista Outro item de lista </pre> | <ul style="list-style-type: none">Um item de listaOutro item de lista |
| <pre> Um item de lista Outro item de lista </pre> | <ol style="list-style-type: none">1 Um item de lista2 Outro item de lista |

Criando links

Criar um link é muito simples, basta utilizar `texto com link`. É necessário, também, utilizar o atributo “`href`” que conterá o endereço ao qual o usuário será remetido ao clicar no link. Vejamos:

```
<a href="http://www.globo.com">Aqui tem um link para o site Globo.com</a>
<a href="pagina2.htm">Aqui um link para a pagina 2</a>
<a href="subdiretorio/pagina2.htm">Aqui tem um link para a página 2</a>
<a href="../pagina1.htm">Aqui um link para a pagina 1</a>
```

Podemos também criar links internos, dentro da própria página - por exemplo, uma tabela de conteúdos ou índice com links para cada um dos capítulos dā página. Tudo o que você precisa é usar o atributo `id` e o símbolo “`#`”. Esse recurso é chamado também de âncora.

Use o atributo `id` para marcar uma área de link:

```
<h1 id="heading1">Cabeçalho 1</h1>
```

E chame o link da seguinte forma:

```
<a href="#heading1">Link para o cabeçalho 1</a>
```

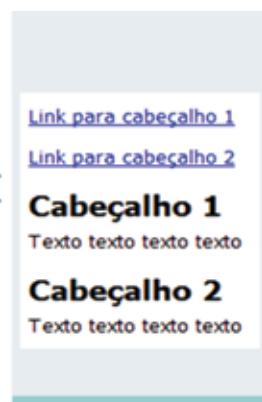
Figura 10 - Exemplo de link interno (âncora)

```
<html>
<head>
    <title> Título da página </title>
</head>

<body>
    <p><a href="#heading1">Link para cabeçalho 1</a></p>
    <p><a href="#heading2">Link para cabeçalho 2</a></p>

    <h1 id="heading1">Cabeçalho 1</h1>
    <p>Texto texto texto</p>

    <h1 id="heading2">Cabeçalho 2</h1>
    <p>Texto texto texto</p>
</body>
</html>
```



Link para e-mail, ou seja, ao clicar será aberta a tela do programa de correio eletrônico, por exemplo, o Microsoft Outlook (caso este esteja instalado na máquina). Veja como fazer:

```
<a href="mailto:flavio@flavioizo.com">E-mail para Flávio</a>
```

Atributo target:

O atributo “target” possibilita abrir o link de diversas formas:

- Para abrir na mesma página, usa-se `target="_self"`;
- para abrir em uma nova página, usa-se `target="_blank"`;
- “`_parent`” e “`_top`” são muito utilizados quando utilizamos frames, por isso não iremos abordá-los neste momento.

`Link para a página 2` ou

`Link para a página 2`

Indicações

Para saber mais sobre as *tags* do HTML, acesse o site <http://pt-br.html.net/> ou pesquise em livros, como o “Use a Cabeça HTML com CSS e XHTML”, da coleção “Use a cabeça”, da editora Alta Books.

Para saber mais sobre a influência das cores nos sentimentos do ser humano, procure livros que discorram acerca dos temas “psicologia das cores” e “teoria das cores”.

Atividades

1. O nosso objetivo será criar o primeiro site, e para isso vamos utilizar os conceitos aprendidos neste capítulo. Assim, crie três páginas:

index.htm

Essa página deverá ter uma mensagem de boas-vindas para o usuário e conter links para as seguintes páginas: index.htm, clientes.htm, produtos.htm, Cead do Ifes.

clientes.htm

Essa página deverá ter nome de alguns clientes (invente alguns nomes) e conter links para as seguintes páginas: index.htm, clientes.htm, produtos.htm, Cead do Ifes.

produtos.htm

Essa página deverá ter nome de alguns produtos (invente alguns nomes) e conter links para as seguintes páginas: index.htm, clientes.htm, produtos.htm, Cead do Ifes.

OBS.: Use sua imaginação e coloque cores, textos com tamanhos diferentes etc.



Anotações

2.2 DIFERENÇA ENTRE JAVASCRIPT E JAVA

Muita gente confunde javascript com java. Primeiramente, precisamos saber: “Java não tem nada a ver com JavaScript”. As diferenças são grandes, e abordaremos aqui as principais.

Em relação ao compilador - Para programar em Java, necessitamos de um Kit de desenvolvimento e um compilador. Entretanto, o Javascript não é uma linguagem que necessite que seus programas se compilem, e sim que estes sejam interpretados pelo navegador quando este lê a página. Lembre-se de que Javascript funciona no lado do cliente (Client-Side) e java funciona no lado do servidor(Server-Side).

Em relação à Orientação a Objetos - Java é uma linguagem de programação totalmente orientada a objetos. Já o Javascript é orientado a objetos, mas sem utilizar classes. O javascript utiliza protótipos para representar as classes. No entanto, poderemos programar sem necessidade dos conceitos OO, tal como se realiza nas linguagens de programação estruturadas como C ou Pascal.

Quanto ao funcionamento - Java é mais potente que Javascript, isto é devido ao fato de que Java é uma linguagem com um escopo mais abrangente, o que permite que façamos aplicações diversas. Entretanto, o Javascript somente permite escrever programas mais simples, que devem ser executados em páginas web.

Quanto à conexão ao banco de dados - O javascript não conecta a banco de dados, tais como Oracle, SQL Server, Postgres, MySql, etc. Isso se dá por um motivo simples (e que já discutimos bastante até aqui): o Banco de dados funciona no lado do Servidor e o javascript no lado Cliente.

Façamos um resumo das diferenças:

| Características | Java | Javascript |
|----------------------------|-------------------|------------------------------|
| Tipo de execução | Compilador | Interpretador |
| Orientação à Objeto | Sim | Sim, mas não possui classes. |
| Funcionamento | Escopo Abrangente | Navegador Web |
| Conecta a BD | Sim | Não |

2.3 INICIANDO COM O JAVASCRIPT

O código javascript surgiu pelo navegador Natscape. Seu nome passou por duas mudanças: primeiramente foi chamado de “Mocha”, logo em seguida virou “LiveScript”, e posteriormente virou o javaScript que nós utilizamos até os dias atuais.

Historicamente há um indício de que o nome javascript foi o escolhido para o lançamento do Netscape 2.0 em 1995 devido à popularização que o Java tinha no momento. Isso soa como uma estratégia de marketing da Netscape.

As estruturas principais do javascript são:

<SCRIPT LANGUAGE=“Javascript”>Código em js</SCRIPT>

ou

<SCRIPT LANGUAGE=“Javascript” SRC=“arquivo.js”></SCRIPT>

Na primeira estrutura, o código js é inserido diretamente entre as *tags* <script>. Já no segundo modelo, o código js é inserido em um arquivo à parte (arquivo.js) e esse arquivo está sendo referenciado (importado) na página.

Normalmente esses códigos são colocados entre as *tags* <HEAD></HEAD> da sua página, embora você possa colocá-los em qualquer lugar da página. Veja na figura 11.

Figura 11 - Exemplo de página chamando código javascript

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Título da minha página</title>
    <script type="text/javascript" language="Javascript" src="arquivo.js"></script>
  </head>
  <body>
    Conteúdo da página do meu site
  </body>
</html>
```

2.4 DOM (*DOCUMENT OBJECT MODEL*)

O javascript possui vários objetos, os quais contribuem para a execução de diversas funcionalidades do js. Para que conheçamos sobre objetos, é extremamente necessário que façamos o estudo do DOM (*Document Object Model*), que nos proporciona modificar desde coisas simples às mais complexas. Portanto, o DOM nos permite alterar tudo (que está no lado cliente) na nossa página WEB.

A estrutura de uma página web respeita uma hierarquia de objetos, que veremos na figura 12. Repare que o objeto “window” é o pai de todos os objetos.

Figura 12 - Hierarquia de objetos em javascript

```
window(objeto pai)
--->document
    --->forms(text, textarea, radio, button, checkbox, select,
              select multiple, file, submit, reset)
    --->links
    --->images
    --->applets
    --->embeds
    --->plugins
    --->anchors
--->frames(coleção de objetos document)
--->navigator
--->location
--->history
--->event
--->screen
```

Abaixo do objeto pai, existem os objetos filhos. A hierarquia de objetos é composta por diversos elementos.

Para representar melhor essa hierarquia, poderíamos fazer uma analogia com a seguinte situação: “Imagine que você queira pintar a porta da sala

de sua casa na cor branca". Se pudéssemos representar em formato de programação, ficaria assim:

```
casa.sala.porta.cor = "branca";
```

Observe que tivemos que acessar casa, depois sala, depois porta e informar que a cor seria branca. O ponto (.) serve para separar os elementos da hierarquia. No javascript funciona da mesma maneira.

Vamos colocar o DOM em prática:

Imagine que você deseja alterar o título da página web aberta. Para isso você vai pedir que o usuário digite um novo título para a página e pedir que o usuário digite o nome dele. Logo em seguida você vai modificar o título da página para o novo título + o nome do usuário. Veja:

Figura 13 - Exemplo de uso do DOM

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Título da minha página</title>
  </head>
  <body>
    <script type="text/javascript" LANGUAGE="JavaScript">
      window.alert("Atualmente o título da página é: " + document.title);
      var TituloNovo = window.prompt("Digite um novo Título para a página?", "Novo Título");
      var Nome = window.prompt("Digite o seu nome!", "Seu Nome");
      var Titulo = TituloNovo + " | Seu nome: " + Nome;
      document.title = Titulo;
    </script>
    Conteúdo da página do meu site
  </body>
</html>
```

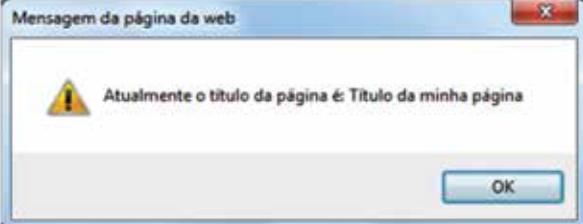
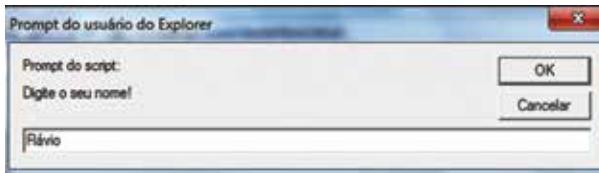
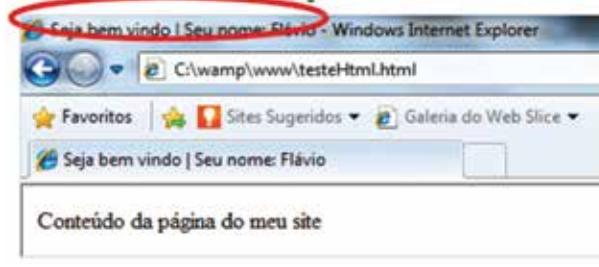
Observe que utilizamos na linha 8 os comandos "`window.alert()`" e "`document.title`". Assim, o primeiro é uma função interna do `window` que serve para exibir um conteúdo em forma de caixa de texto. Já o segundo serve para recuperar o título da página web.

Atenção



Como o `window` é o objeto pai de todos, por padrão não é necessário utilizar o nome `window` antes dos elementos hierarquicamente inferiores. No nosso exemplo, nós utilizamos no `alert` e não utilizamos no `document`.

Ao exibir a página no navegador vamos obter as seguintes saídas:

| | |
|--|---|
|  | Primeira tela exibindo o título atual da página. |
|  | Tela solicitando a entrada de um novo título para a página. |
|  | Tela solicitando a entrada do nome do usuário. |
|  | Em destaque o novo título da página. |

Como percebemos na figura 12, existem diversos elementos que fazem parte do DOM. Vamos citar alguns exemplos de elementos no quadro abaixo:

| Objeto | Descrição |
|--|---|
| <code>window.alert("texto");</code> | Exibe uma janela de alerta no estilo popup |
| <code>window.location.href="url_aqui";</code> | Redireciona a página para a url informada |
| <code>window.open("Pag1.htm","janela","height = 300, width = 400");</code> | Abre uma nova página no estilo "popup". Onde, "janela" é o nome da janela; "height" é a altura da janela e "width" é a largura da janela. |
| <code>window.close();</code> | Fechas a janela aberta no navegador |
| <code>window.print();</code> | Abre uma janela de impressão. |
| <code>window.history.back();</code> | Volta para a página anterior. |
| <code>window.history.forward();</code> | Avança para a próxima página. OBS.: Para avançar é preciso ter voltado uma pagina antes. |

| Objeto | Descrição |
|--|--|
| <code>window.history.go(valor);</code> | Retorna ou avança para uma determinada página. Onde valor: (-1) retorna uma página; (-2) retorna duas páginas; (1) avança uma página; (2) avança duas páginas; e assim por diante. |
| <code>window.document.write("texto");</code> | Escreve um texto na tela. Onde "texto" é o texto a ser escrito. |
| <code>window.navigator.appName;</code> | Mostra o nome do navegador que está sendo utilizado pelo usuário. Ex.: "Microsoft Internet Explorer"; "Netscape" etc. |
| <code>window.navigator.appVersion;</code> | Exibe a versão do navegador. |
| <code>window.navigator.platform;</code> | Exibe o tipo de máquina que está sendo utilizada. Ex.: Win32, 'MacPPC' etc. |
| <code>window.document.URL;</code> | Exibe a URL da página |
| <code>window.confirm("texto");</code> | Tela com os botões OK e Cancelar. Onde "texto" é o texto que vai aparecer na janela da mensagem. |
| <code>window.prompt("msg","texto");</code> | Abre uma tela pedindo para o usuário inserir um valor. Onde: "msg" é a mensagem indicando o que o usuário deve digitar; e "texto" é o texto que aparece dentro do campo. |

Indicações

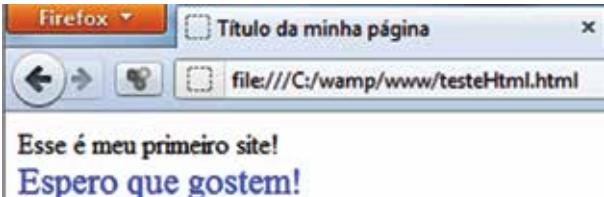


Para saber mais sobre os outros elementos que vimos na figura, você deve fazer uma pesquisa nos livros indicados na referência bibliográfica ou na internet.

2.5 INSERINDO UM TEXTO NA PÁGINA WEB

Para inserir um texto na web, basta digitar o texto entre as tags <body> e </body>.

Figura 14 - Código e como será exibido o site



```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Título da minha página</title>
  </head>
  <body>
    Esse é meu primeiro site!<br />
    <font style="font-size: 12pt; color: #0000ff; font-family: Verdana, Arial">Espero que gostem!</font>
  </body>
</html>
```

Observe que no segundo texto nós utilizamos um recurso chamado CSS (*Cascading Style Sheets*, ou Folha de Estilos em Cascata).

Atualmente, todas as páginas webs utilizam o recurso do CSS. Como nosso foco não é aprofundar neste recurso, sugiro que você aproveite e aperfeiçoe seus conhecimentos lendo a respeito.

Uma dica é acessar o site <http://www.w3schools.com/css/> e ler o tutorial disponibilizado para aprendizagem.



Indicações

Outra maneira de escrever um texto na página web é utilizando o javascript. Basta utilizar o método “*write*” visto no capítulo anterior:

```
window.document.write("texto");
```

2.6 FORMATAÇÃO DOS CARACTERES

Para formatar um texto, podemos utilizar os conceitos de CSS, como vimos na figura 14.

Outras tags que podemos utilizar são:

| | |
|---|---|
| <code><p>texto</p></code> | Indica parágrafo |
| <code><PRE>texto</PRE></code> | Mantém o texto na forma como está sendo escrito. Ex.: Só gera quebra de linha se apertar "Enter". |
| <code>texto</code> | Aplica o estilo negrito. |
| <code><I>texto</I></code> | Aplica o estilo itálico. |
| <code><U>texto</U></code> | Aplica o estilo sublinhado. |

É importante salientar que todas as tags acima aceitam que se use o atributo “style” para aplicar um estilo CSS. Vejamos na figura 15:

Figura 15 - Formatando o texto que será exibido no site

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Titolo da minha página</title>
  </head>
  <body>
    Esse é meu primeiro site!<br />
    <font style="font-size: 12pt; color: #0000ff; font-family: Verdana">Texto A</font>
    <p style="font-size: 14pt; color: fuchsia; font-family: times, serif">Texto B</p>
    <pre style="font-size: 10pt; color: red; font-family: Helvetica, Arial">Texto C</pre>
  </body>
</html>

```

2.7 FORMATAÇÃO DO DOCUMENT

O objeto `document` é o objeto que contém todas as informações sobre a página atual (documento). Para corroborar com essa informação, esse objeto possui propriedades que possibilitam a configuração (formatação) da página. Vejamos algumas dessas propriedades:

| Propriedades da página | |
|--|---------------------------------------|
| <code>document.alinkColor = "blue";</code> | Informa a cor do link ativo da página |
| <code>document.bgColor = "aqua";</code> | Informa a cor de fundo da página |
| <code>document.fgColor = "red";</code> | Informa a cor do texto da página |

| Propriedades da página | |
|--|--|
| document.lastModified; | Informa a hora e a data da última modificação da página. |
| document.linkColor = "red"; | Informa a cor padrão dos links. |
| document.title; | Informa o título da página. |
| document.URL; | Informa a URL da página. |
| document.vlinkColor = "orange"; | Informa a cor dos links que já foram visitados. |

OBS.: Onde nós utilizamos nome das cores, você pode substituir pelo valor das cores em Hexadecimal, pois poderá utilizar uma gama muito maior de opções de cores.

2.8 VARIÁVEIS

As variáveis são extremamente importantes para os sistemas, independentemente se os sistemas são web ou não. Variáveis são espaços reservados na memória do computador para que possamos guardar algum valor (dado) durante a execução dos nossos scripts (códigos de programação).

Assim como na maioria das linguagens de programação, no javascript as variáveis são declaradas utilizando-se a palavra reservada “var”.

Figura 16 - Exemplo de uso das variáveis

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Titulo da minha página</title>
  </head>
  <body>
    <script type="text/javascript" language="JavaScript">
      var idade = 29;
      var nome = "Flávio";
      document.write("Meu nome é "+nome+" e tenho "+idade+" anos!");
    </script>
  </body>
</html>
```

Se abrirmos a página com o código da figura 16, teremos como retorno a seguinte saída: “Meu nome é Flávio e tenho 29 anos!”.

Você deve ter percebido que nós não fornecemos o tipo da variável (ex.: integer, char, boolean etc.). Isso se deve ao fato de que as variáveis em javascript não são tipadas, ao contrário de linguagens como Java e Visual Basic. Sendo assim, basta que você insira o valor e a variável

automaticamente interpretará o tipo de conteúdo que será armazenado na variável.

Atenção

Para converter valores de String para integer ou float basta utilizar as funções “parseInt()” ou “parseFloat()”, respectivamente.

Um ponto importante que deve ser destacado é que as variáveis utilizadas estarão disponíveis durante a execução da página, ou seja, se você acessar outra página ou fechá-la, as variáveis deixarão de existir.

2.9 NOMES RESERVADOS (PALAVRAS RESERVADAS)

Assim como em qualquer linguagem, o javascript possui palavras reservadas. Essas palavras são reservadas para que executem algum tipo de comando. Sendo assim, não podemos utilizá-las durante a execução de nosso *script* para outra coisa que não para o motivo pelo qual foi criada.

Vejamos no quadro abaixo algumas das palavras reservadas no javascript:

| | | | |
|----------|------------|-------------|--------------|
| abstract | eval | long | return |
| alert | export | math | setInterval |
| array | extends | name | setTimeout |
| blur | false | native | short |
| boolean | final | navigator | static |
| break | finally | new | status |
| byte | float | number | string |
| case | focus | null | super |
| catch | for | object | switch |
| char | function | onBlur | synchronized |
| class | goto | onLoad | this |
| confirm | history | open | throw |
| const | if | outerHeight | throws |
| continue | image | package | transient |
| date | implements | parent | try |
| debugger | import | parseFloat | typeof |
| default | in | parseInt | true |
| delete | instanceof | print | var |
| do | int | private | void |
| document | interface | prompt | volatile |
| double | isNaN | protected | while |
| else | length | public | window |
| enum] | location | RegExp | with |

Uma dica importante é estar sempre atento às novas palavras reservadas que poderão surgir no javascript.

2.10 OPERADORES

Primeiramente, devemos discutir acerca do que são os operadores. Os operadores são símbolos matemáticos que nos permitem trabalhar com valores.

Os operadores são divididos em operadores aritméticos, operadores relacionais, operadores de atribuição e operadores lógicos.

Operadores aritméticos:

Esses tipos de operadores nos auxiliam na resolução de cálculos envolvendo expressões aritméticas. Vejamos:

| Operadores | Matemática | Javascript |
|---------------|---------------|-------------|
| Adição | $A + B$ | $A + B$ |
| Subtração | $A - B$ | $A - B$ |
| Multiplicação | $A \times B$ | $A * B$ |
| Divisão | $A \div B$ | A / B |
| Módulo (mod) | | $A \% B$ |
| Potenciação | A^B | $A ^ B$ |
| Radiciação | $\sqrt[B]{A}$ | $A ^ (1/B)$ |

Operadores relacionais (de comparação):

Esses tipos de operadores nos auxiliam na resolução de cálculos envolvendo expressões relacionais. Devemos lembrar que os operandos da relação devem ser do mesmo tipo e o resultado sempre será lógico (verdadeiro ou falso). Vejamos:

| Operadores | Matemática | Javascript |
|------------------|------------|------------|
| Igual a | $A = B$ | $A == B$ |
| Diferente de | $A \neq B$ | $A != B$ |
| Maior que | $A > B$ | $A > B$ |
| Menor que | $A < B$ | $A < B$ |
| Maior ou igual a | $A \geq B$ | $A >= B$ |
| Menor ou igual a | $A \leq B$ | $A <= B$ |

Operadores de atribuição:

Esses tipos de operadores nos auxiliam na atribuição de valores às variáveis. Vejamos:

| Operadores | Significado | Javascript |
|-----------------|----------------|---------------------|
| <code>+=</code> | $A = A + B$ | <code>A += B</code> |
| <code>-=</code> | $A = A - B$ | <code>A -= B</code> |
| <code>*=</code> | $A = A * B$ | <code>A *= B</code> |
| <code>/=</code> | $A = A \div B$ | <code>A /= B</code> |
| <code>%=</code> | $A = A \% B$ | <code>A %= B</code> |
| <code>++</code> | $A = A + 1$ | <code>A++</code> |
| <code>--</code> | $A = A - 1$ | <code>A--</code> |

Operadores lógicos:

Esses tipos de operadores nos auxiliam nos testes lógicos. Vejamos:

| Operadores | Significado | Javascript, Onde A=9 e B=4 |
|-------------------------|-------------|---|
| <code>!</code> | NÃO | $(A != 5)$ obtém-se "true" |
| <code> </code> | OU | $(A < 5) \text{ OU } (B > 3)$ obtém-se "true" |
| <code>&&</code> | E | $(A < 5) \text{ E } (B > 3)$ obtém-se "false" |

2.11 FUNÇÕES

Toda linguagem de programação possui um recurso que é extremamente importante para a organização e reaproveitamento do código: as funções.

As funções são partes específicas do código e são chamadas através do seu próprio nome. As funções javascript podem ser chamadas através de botões, de links ou por meio de outras funções.

A sintaxe da função é:

Figura 17 - Exemplo de uso de funções

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Título da minha página</title>
    <script type="text/javascript" language="JavaScript">
      function Somar(n1,n2){
        var soma;
        soma = n1 + n2;
        return soma;
      }
    </script>
  </head>
  <body>
    <script type="text/javascript" language="JavaScript">
      var n1 = 3;
      var n2 = 6;
      var soma = Somar(n1,n2);
      document.write("Resultado da Soma: " + soma);
    </script>
  </body>
</html>
```

Perceba os seguintes pontos:

- para começar uma função, nós devemos utilizar a palavra reservada “function”;
- a função deve ser declarada dentro das tags <HEAD> e </HEAD>;
- o “return” foi utilizado porque essa função ao ser executada retorna valor para uma variável. Caso não retornasse valor algum, deveríamos simplesmente não coloca-lo (ideologia de procedimento).

Caso quiséssemos chamar a função através de um link:

```
<a href="Javascript:Somar(3,6);">Clique aqui para somar 3 com 6</a>
```

E caso quiséssemos chamar a função através de um botão de formulário:

```
<form name="formulario">
  <input type="button" value="Somar 3 com 6" onClick="alert(Somar(3,6));" />
</form>
```

- 1) Para saber mais sobre os itens de formulário, veja o tópico “formulários”. E para saber sobre os eventos, veja o tópico “Eventos”.



Atenção

Atenção

2) Existem funções internas do javascript, ou seja, funções que já vêm prontas na linguagem. Essas funções são denominadas “Built-in”.

3) Para não bagunçar o seu código, você também pode colocar todas as funções em um único arquivo com extensão .js e chamá-lo através do comando:

```
<script type="text/javascript" src="funcoes.js"></script>
```

2.12 EVENTOS

Todos os elementos vistos no tópico 2.4 podem ser chamados pelos eventos denominados por nós.

Os eventos são recursos que chamam funções ao serem executados. Eles são muito utilizados para facilitar a execução de tarefas pelo usuário. Vejamos alguns eventos:

| Evento | Significado | Funcionamento |
|-----------------------|------------------------------------|---|
| OnClick(); | Ao clicar sobre | Todos os elementos |
| OnChange(); | Ao alterar o valor atual | <select>, <input> e <textarea> |
| OnBlur(); | Ao clicar fora do objeto | <a>, <input>, <select>, <textarea> etc. |
| OnLoad(); | Ao abrir o site | <body> e <frameset> |
| OnUnload(); | Ao fechar o site | <body> e <frameset> |
| OnMouseOver(); | Ao passar o mouse em cima | Todos os elementos |
| OnMouseMove | Move o mouse | Todos os elementos |
| OnMouseOut(); | Ao tirar o mouse de cima | Todos os elementos |
| OnFocus(); | Ao colocar o cursor sobre o objeto | <a>, <input>, <select>, <textarea> etc. |
| OnSubmit(); | Ao submeter (enviar) o formulário | <form> |
| OnKeyUp(); | Ao soltar a tecla | <input>, <select>, <textarea> |
| OnKeyDown(); | Ao pressionar a tecla | <input>, <select>, <textarea> |
| OnKeyPress(); | Ao pressionar e soltar a tecla | <input>, <select>, <textarea> |

Todos os eventos citados podem ser chamados por links, botões de formulário ou dentro das tags do html. Abaixo vamos citar alguns exemplos de eventos sendo chamados:

```
<input type="button" value="Imprimir" onClick="imprimir()" />
<BODY onLoad="bemVindo();">
<input type="text" name="numero" value="" onFocus="alerta();">
<a href="principal.html" onMouseOut="alerta();">Principal</a>
<BODY onUnload="alert('Obrigado!');">
```

Você pode acionar um evento para que uma figura seja trocada em tempo de execução. Para isso, basta fazer uma combinação de eventos com o link da imagem. Veja:

```
<a href="principal.html" onMouseOver="document.images['figura'].src = 'figura2.jpg';"
onMouseOut="document.images['figura'].src = 'figura1.jpg';">
![Logo](figura1.jpg)

```

Esse efeito é muito bacana e faz com que o seu site tenha um visual diferente!



Atenção

Agora que você já viu sobre o funcionamento dos eventos, basta utilizar a criatividade e implementar os scripts no seu código.

2.13 ESTRUTURA CONDICIONAL

A estrutura condicional está presente na maioria das linguagens de programação. Sua condição deve sempre retornar um valor lógico: “sim” ou “não”. Portanto, as condições servem para que um bloco de comando seja executado somente SE a condição for satisfeita. Veja o exemplo:

Figura 18 - Estrutura condicional

```
<script type="text/javascript" language="JavaScript">
var n1 = 3;
var n2 = 6;
var opcao = window.prompt("Digite + ou -, "+"")
if (opcao == "+"){
    document.write("Resultado da Soma: " + eval(n1 + n2));
} else {
    if (opcao == "-"){
        document.write("Resultado da Subtração: " + eval(n1 - n2));
    }
}
</script>
```

Na figura 18, recebemos 3 para a variável “n1” e 6 para a variável “n2”. Pedimos para o usuário digitar a opção desejada (+ ou menos) e armazenamos na variável “opção”. Em seguida, testamos qual das opções o usuário digitou. Se for “+”, será impressa a soma dos valores; se for “-”, será impressa a subtração dos valores.

Devemos lembrar que o “*else*” (senão) foi útil nos nosso exemplo, pois ele só testará se o usuário digitou “-” se o teste (*opcao == “+”*) for falso.

2.14 FORMULÁRIOS

O formulário é o meio que permite criar mais interatividade entre a programação e o usuário. É por meio dele que recebemos os valores inseridos pelos usuários e podemos armazená-los em um banco de dados, testá-los com estruturas condicionais etc.

Uma estrutura de formulário possui vários campos que permitem a entrada de dados. Abaixo, no quadro, colocamos algumas tags utilizadas nos formulários.

| TAG | Descrição |
|------------|---|
| <form> | Define um formulário |
| <input> | Define um campo de entrada de dados |
| <textarea> | Define um text-area (um campo de texto de diversas linhas) |
| <label> | Define a label to a control |
| <select> | Define uma lista selecionável selectable list (também chamada de drop-down box) |
| <option> | Define uma opção drop-down box |
| <button> | Define um botão |

A seguir, falaremos mais sobre cada uma das *tags* utilizadas nos formulários.

Tag <form>

Os *forms* fazem parte do código *html*, portanto devem ser definidos como as *tags* desta linguagem. Sempre com uma *tag* de abertura e outra de fechamento, inserindo o conteúdo do formulário entre estas duas. Veja a seguir:

```
<!-- tag de abertura do formulário -->
<form name="form1" action="pagina.php"
      method="post" enctype="application/x-www-form-urlencoded">

    <!-- campos do formulário -->

</form> <!-- tag de fechamento do formulário -->
```

Parâmetros da tag form

- **Name:** Nome do formulário. Não pode existir mais de um formulário com o mesmo valor para o “name” na mesma página.
- **Action:** Todo formulário deve ser enviado para algum local. Sendo assim, o action é o script ou página para onde os dados serão submetidos. É neste script que normalmente os dados são tratados.
- **Method:** É o método de envio dos dados. Pode ter dois valores:
GET = Passa os valores pela URL, ou seja, podemos ver as variáveis e os seus respectivos valores na URL da página destino, definida no campo *action*. Tem limitação de 256 bytes. Não é muito aconselhável o uso do método GET, pois ele expõe o nome e valor das variáveis, o que pode ser utilizado por pessoas com intenções maléficas.
POST = Passa as variáveis de maneira transparente para o usuário. É o método mais aconselhável, pois não fica visível aos olhos do usuário.
- **Enctype:** é o tipo de empacotamento de envio dos dados do formulário. O valor padrão é o “application/x-www-form-urlencoded”, no entanto, quando desejarmos fazer envio de arquivo pelo formulário (campo “input file”), nós devemos definir o enctype como “multipart/form-data”, pois este irá converter o arquivo para binário e enviá-lo.

Tag <label>

Essa tag define um rótulo para o campo. Exemplo:

Figura 19 - Label com evento

```
<label onmouseover="alert('Teste');" >Rótulo
    <!-- campos do formulário -->
</label>
```

Observe que, no exemplo acima, eu combinei o rótulo com o evento de javascript “onmouseover()”. Assim, ao passar o mouse por cima do rótulo, será exibido um alert() com a mensagem “teste”;

Tag <input>

A tag input serve para efetuar a entrada de dados. Ela possui vários tipos (*type*) e por isso é muito utilizada. Os *types* existentes são:

- text, password, file, hidden, checkbox, radio, button, reset, submit e image.

Abaixo, você poderá ver os *inputs* sendo utilizados na prática:

Figura 20 - Tipos de inputs e seus atributos

```
<form id="form1" name="form1" method="post" action="" enctype="multipart/form-data">
    <label>Campo Texto
        <input type="text" name="texto" value="" size="15" accesskey="a" tabindex="1" />
    </label><br />
    <label>Campo Senha
        <input type="password" name="senha" value="" size="15" accesskey="b" tabindex="2" />
    </label><br />
    <label>Campo Arquivo
        <input type="file" name="arquivo" value="" size="15" accesskey="c" tabindex="3" />
    </label><br />
    <label>Campo Oculto
        <input type="hidden" name="oculto" value="cod01" />
    </label><br />
    <label>Campo de checagem
        <input type="checkbox" name="check01" value="livro_01" accesskey="d" tabindex="4" />
    </label><br />
    <label>Feminino
        <input type="radio" name="sexo" value="Feminino" accesskey="e" tabindex="5" />
    </label>
    <label>Masculino
        <input type="radio" name="sexo" value="Masculino" accesskey="f" tabindex="6" />
    </label><br />
    <input type="button" name="button01" value="Botão_01" accesskey="g" tabindex="7" />
    <input type="reset" name="apagar" value="Apagar" accesskey="h" tabindex="8" />
    <input type="submit" name="enviar" value="Enviar" accesskey="i" tabindex="9" />
    <input type="image" name="btImg" src="enviar.gif" border="1" accesskey="j" tabindex="10" />
</form>
```

Onde:

- **Type:** é o tipo de input que está sendo utilizado
- **Name:** atribui um nome para o campo
- **Id:** é o atributo de identificação para as tags do formulário e do html de modo geral. É muito utilizado para receber valores no modo client-side. Já no modo server-side, o atributo de identificação utilizado é o name.
- **Value:** atribui um valor padrão para o campo. É muito utilizado no inputs do tipo hidden, radio, checkbox. Nos botões do tipo submit, button e reset, o valor definido em “value” é o texto que vai aparecer para o usuário.

- **Size:** define o tamanho do campo em caracteres.
- **Maxlength:** define a quantidade máxima de caracteres do campo.
- **Accesskey:** define a tecla de acesso rápido (sem uso do mouse) para o campo. Lembre-se de que você deve utilizar a seguinte combinação: SHIFT + ALT + accesskey.
- **Tabindex:** define a sequência para que o campo receba o foco do cursor. Ex.: Ao apertar a tecla TAB, o primeiro campo será o que tiver o tabindex igual a 1. Ao apertar novamente, ele irá para o tabindex 2 e assim sucessivamente.
- **Src:** é o local onde o HTML irá procurar a imagem que vai aparecer no input do tipo “image”;
- **Border:** ativa(1) ou desativa(0) a borda no input do tipo “image”;
- **Checked:** utilizado em campos do tipo “radio” e “checkbox”, essa opção marca o campo, ou seja, o formulário já abre com a opção selecionada.
- **Disabled:** utilizado em todos os campos, esse atributo desabilita o campo. Não é possível nem enviar o valor dele quando o formulário é submetido.
- **Readonly:** torna o campo como somente leitura. Não é possível alterar o valor para do campo.

Observe que eu coloquei dois campos do tipo “radio” e esses possuem o mesmo nome. Isso foi feito porque o usuário não poderá acionar as duas opções relacionadas a sexo, ele só poderá selecionar uma. Assim, quando desejarmos que o usuário selecione somente uma opção dentro do bloco de opções do radio, basta que as coloquemos com o mesmo nome.

Outra observação importante é que, ao tentar digitar a senha, você vai perceber que aparecerão * ou •, de acordo com o navegador, isso porque o tipo de input é password.

Veja o resultado da interpretação do código da figura 21:

Figura 21 - Tipos de inputs e seus atributos

The screenshot shows a web browser window with the URL `localhost/testes/formulario.php`. The form contains the following elements:

- Campo Texto:** A text input field.
- Campo Senha:** A password input field.
- Campo Arquivo:** A file input field with a "Selecionar arquivo..." button.
- Campo Oculto:** An hidden input field.
- Campo de checagem:** A checkbox input field.
- Feminino / Masculino:** Radio buttons for gender selection, with "Masculino" selected.
- Botão 01 / Apagar / Enviar / Enviar:** Buttons for submission and cancellation.

Tag <textarea>

O textarea (ou área de texto) é o campo que exibe um campo de texto maior, que permite ao usuário inserir um texto com várias linhas.

Figura 22 - Exemplo de textarea

```
<label>Mensagem
<textarea name="area1" rows="5" cols="20" accesskey="a" tabindex="1">
    <!-- Conteúdo da área de texto --&gt;
&lt;/textarea&gt;
&lt;/label&gt;</pre>
```

Onde:

- **Name:** atribui um nome para o campo.
- **Rows:** define o número de linhas do campo. Esse atributo somente limita o tamanho do campo e não a quantidade de valores inseridos nele.
- **Cols:** define o número de colunas do campo (largura).
- **Disabled:** esse atributo desabilita o campo. Não é possível nem enviar o valor dele quando o formulário é submetido.
- **Readonly:** torna o campo como somente leitura. Não é possível alterar o valor para do campo.
- **Accesskey:** define a tecla de acesso rápido (sem uso do mouse) para o campo. Lembre-se de que você deve utilizar a seguinte combinação: SHIFT + ALT + accesskey.
- **TabIndex:** define a sequência para que o campo receba o foco do cursor. Ex.: Ao apertar a tecla TAB, o primeiro campo será o que tiver o tabindex igual a 1. Ao apertar novamente, ele irá para o tabindex 2 e assim sucessivamente.

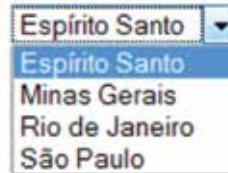
Tag <select>

Essa opção de conteúdo é muito utilizada e importante quando necessitamos exibir informações em forma de lista. Essa lista pode ser exibida em forma de **listagem** ou através de um **combo**. Veja os dois exemplos:

Tipo Combo (você clica na seta e aparecem as opções):

Figura 23 - Exemplo de select do tipo “combo”

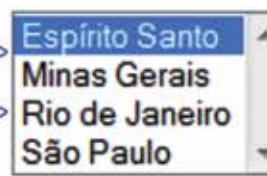
```
<select name="selecao1">
    <option value="ES">Espírito Santo</option>
    <option value="MG">Minas Gerais</option>
    <option value="RJ">Rio de Janeiro</option>
    <option value="SP">São Paulo</option>
</select>
```



Tipo Listagem (as opções já aparecem listadas de acordo com o número de linhas estabelecidas):

Figura 24 -: Exemplo de select do tipo “combo”

```
<select name="selecao1" size="4">
    <option value="ES">Espírito Santo</option>
    <option value="MG">Minas Gerais</option>
    <option value="RJ">Rio de Janeiro</option>
    <option value="SP">São Paulo</option>
</select>
```



Observe que o valor contido em “value” é o valor que será enviado pelo formulário. Existem ainda outros atributos que podem ser utilizados:

- Multiple:** permite ao usuário escolher mais de uma opção. Para que essa opção funcione, você deve estabelecer o “size” com um valor maior que 1 (um).
- Selected:** esse atributo deve ser colocado na tag <option> e faz com o item da lista já apareça selecionado.

Você deve lembrar que, para selecionar mais de uma opção no modo MULTIPLE, você deve segurar a tecla CTRL ou SHIFT.



Atenção

Para aprimorar seus conhecimentos, você deve procurar ler sobre as tag:

<optgroup>, <fieldset> e <legend>



Indicações

Vejamos um exemplo de um formulário, inclusive fazendo-se uso de tabelas:

Figura 25 - Arquivo formulario_completo.html

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Envio de Formulário</title>
5     <script type="text/javascript" src="funcoes.js"></script>
6   </head>
7   <body>
8     <form name="form1" enctype="application/x-www-form-urlencoded" method="post"
9       action="procForm.html" onsubmit="return(fn_validar(this));">
10       <table width="500" border="1" cellspacing="0" cellpadding="2">
11         <tr>
12           <td colspan="2" align="center">Formulário de Cadastro</td>
13         </tr>
14         <tr>
15           <td width="30%">Nome:</td>
16           <td width="70%"><input name="nome" type="text" size="40" /></td>
17         </tr>
18         <tr>
19           <td>Endereço:</td>
20           <td><input name="endereco" type="text" size="40" /></td>
21         </tr>
22         <tr>
23           <td>Cidade:</td>
24           <td><label>
25             <input type="text" name="cidade" id="cidade" />
26             <select name="estado">
27               <option>Selecione um estado...</option>
28               <option value="ES">Espírito Santo</option>
29               <option value="MG">Minas Gerais</option>
30               <option value="RJ">Rio de Janeiro</option>
31               <option value="SP">São Paulo</option>
32             </select>
33           </label>
34         </td>
35       </tr>
36       <tr>
37         <td>Sexo:</td>
38         <td>
39           <input type="radio" name="sexo" value="Feminino" /> Feminino
40           <input type="radio" name="sexo" value="Masculino" /> Masculino
41         </td>
42       </tr>
43       <tr>
44         <td>Nascimento:</td>
45         <td><input type="text" name="nascimento" size="10" /></td>
46       </tr>
47       <tr>
48         <td colspan="2" align="center">
49           <input type="submit" name="cadastrar" value="Cadastrar" />
50           <input type="reset" name="apagar" value="Apagar" />
51         </td>
52       </tr>
53     </table>
54   </form>
55 </body>
56 </html>
```

Figura 26 - Arquivo funcoes.js

```
1  function fn_vazio(campo){
2      if (campo.value == ""){
3          return true;
4      }else{
5          return false;
6      }
7  }
8  function fn_numero(campo){
9      if (isNaN(campo.value)){
10         return true;
11     }else{
12         return false;
13     }
14 }
15 function fn_selecionado(campo){
16     if (campo.selectedIndex == "") {
17         return false;
18     }else{
19         return true;
20     }
21 }
22 function fn_checkado(campo){
23     if ((campo[0].checked) || (campo[1].checked)){
24         return true;
25     }else{
26         return false;
27     }
28 }
29 function fn_validar(form){
30     var erro = "";
31     if (fn_vazio(form.nome)){
32         erro = erro + "\nNome";
33     }
34     if (fn_vazio(form.endereco)){
35         erro = erro + "\nEndereço";
36     }
37     if (fn_vazio(form.cidade)){
38         erro = erro + "\nCidade";
39     }
40     if (!fn_selecionado(form.estado)){
41         erro = erro + "\nEstado";
42     }
43     if (!fn_checkado(form.sexo)){
44         erro = erro + "\nSexo";
45     }
46     if (fn_vazio(form.nascimento)){
47         erro = erro + "\nNascimento";
48     }
49     if (fn_numero(form.nascimento)){
50         erro = erro + "\nNúmero";
51     }
52     if (erro == ""){
53         return true;
54     }else{
55         alert("Campo(s) vazio(s) ou inválido(s): " + erro);
56         return false;
57     }
58 }
```

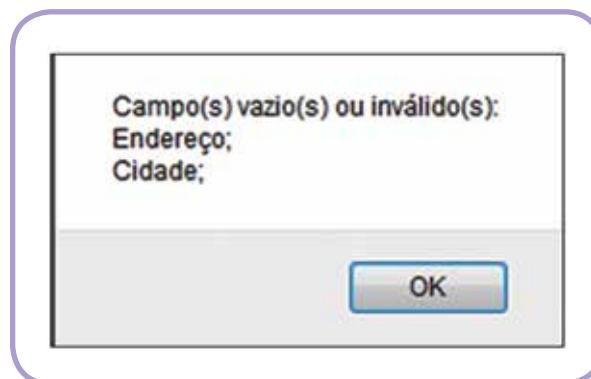
The screenshot shows a web page titled "Formulário de Cadastro". It contains several input fields: "Nome" (Name), "Endereço" (Address), "Cidade" (City) with a dropdown menu showing "Selecionar um estado..." (Select a state...), "Sexo" (Sex) with radio buttons for "Feminino" (Female) and "Masculino" (Male), and "Nascimento" (Birthdate) with a date input field in the format "(DDMMAAAA)". At the bottom are two buttons: "Cadastrar" (Register) and "Apagar" (Delete).

Quando utilizamos um botão do tipo “submit”, isso indica que todos os valores digitados no formulário serão enviados para a página identificada no atributo “action”. Já o botão do tipo “reset”, apaga todos os dados inseridos no formulário.

Aliás, repare bem! Na linha 09 do arquivo `formulario_completo.html` estamos chamando o evento `onsubmit="return(fn_validar(this));"`. A função `fn_validar(this)` está presente dentro do arquivo de funções javascript `funções.js`

De maneira resumida, o que acontece é que, ao clicar no botão “Cadastrar”, o formulário é submetido; porém, antes de efetivamente ser submetido, é chamada a função “`fn_validar(this)`”, que passa o objeto formulário como parâmetro. Assim, caso o retorno seja `true` (não encontrou nenhum erro), o formulário será submetido; caso contrário, a mensagem de erro será exibida em forma de `alert()`.

Figura 28 - Exemplo de mensagem de erro no formulário



Os formulários devem ser validados para saber se uma data é válida, se os campos estão vazios etc.

Pesquise especialmente sobre as expressões regulares.

Pesquise também sobre as funções: isNaN(), parseInt() e parseFloat.



Indicações

2.15 ERROS COMUNS COMETIDOS PELOS PROGRAMADORES

Tudo o que executamos está susceptível a erros, e no javascript não é diferente, os erros podem acontecer, sim! Existem três situações em que eles ocorrem:

- **Na execução**

O script carrega e não acusa nenhum tipo de erro. No entanto, quando o código é executado (acionado por algum evento) um erro é acusado. Normalmente, esse tipo de erro acontece quando colocamos nomes incorretos de variáveis e funções. Exemplo: Confundimos e chamamos a função “`alerta()`” quando o correto seria “`alert()`”.

- **No carregamento**

Os erros no carregamento acontecem quando nos esquecemos de colocar algum parêntese ou chave. Isso impede o carregamento correto do código javascript na memória. Exemplo: chamamos a função `alert("Bem vindo!");`

Perceba que esquecemos de fechar as aspas.

- **Durante a criação do código (erros lógicos)**

Esse tipo de erro acontece quando nosso script executa, porém não da forma como esperávamos. Sendo assim, devemos analisar minuciosamente o código, pois o erro não é de sintaxe da linguagem e sim da forma como criamos a lógica de programação do código.

Uma dica para descobrir onde está o “erro” é analisando o código e colocando pontos de passagem nos blocos de comando, para imprimir o valor de variáveis ou somente para marcar um local e verificar se tudo está normal. Para isso, pode-se utilizar o “`alert(variável)`” ou “`alert('Passou aqui 1')`”.

Como percebemos, os erros podem acontecer. Então tente evitá-los ao máximo, programando com cautela e atenção.

No entanto, caso perceba que algo está incorreto com o seu código, não hesite em verificar seu javascript seguindo as regras informadas acima.

2.16 CAIXAS DE MENSAGEM

As caixas de mensagem são muito utilizadas e servem para que o programador envie uma mensagem de alerta ao usuário.

Ao longo dos tópicos anteriores, já citamos a função interna (*Built In*) `alert()`.

Vamos relembrar o seu funcionamento:

```
window.alert("Atualmente o título da página é: " + document.title);
```

Indicações



O Consórcio World Wide Web (W3C) é um consórcio internacional no qual organizações filiadas, uma equipe em tempo integral e o público trabalham juntos para desenvolver padrões para a Web. Liderado pelo inventor da web Tim Berners-Lee e o CEO Jeffrey Jaffe, o W3C tem como missão conduzir a World Wide Web para que este atinja todo seu potencial, desenvolvendo protocolos e diretrizes que garantam seu crescimento de longo prazo.

Acesse o site <http://www.w3c.org> ou o escritório brasileiro <http://www.w3c.br> para conhecê-lo.

Atividades



1. Você deve criar uma página chamada `formulario.html` contendo um formulário que contenha os seguintes campos: nome (text), endereço (text), estado (combobox), e-mail (text), telefone (text), sexo (radio) e comentários (textarea).

2. Crie um arquivo chamado `validacao.js` contendo funções que validem todos os campos utilizados no formulário da atividade 1. Lembre-se de fazer a inclusão do arquivo na página `formulario.html` e chamar a função de validação. Caso algum campo não tenha sido preenchido, você deve exibir uma mensagem de “alert” ao estilo “O campo xyz deve ser preenchido!”.



Anotações

LINGUAGEM SERVER-SIDE

Olá!

Neste capítulo três, vamos discorrer acerca da linguagem server-side.

Para abordar os conceitos deste modelo, foi selecionado como parâmetro de aplicação prática a linguagem PHP (PHP: Hypertext Preprocessor), que é uma linguagem de programação interpretada, livre e muito utilizada para gerar conteúdo dinâmico na web.

Partindo dessa abordagem, nós iremos verificar qual é a sintaxe básica do PHP e quais ferramentas poderão ser úteis na sua aprendizagem. Também verificaremos como aplicar essa sintaxe e elaborar um projeto web que seja significativo.

No decorrer deste estudo, falaremos acerca dos formulários para envio de informações entre páginas, um procedimento valioso durante a navegação de um sistema web.

Fecharemos o capítulo falando sobre sessão, que é um recurso extremamente importante em páginas/sistemas que necessitam de autenticação de segurança; e abordaremos, ainda, o upload de arquivos.

Os conhecimentos adquiridos neste capítulo serão úteis para a aprendizagem dos demais conteúdos trabalhados nesta disciplina.

Bons estudos!

Angue risus ac
ne velit at tellus.
massa porttitor
sectetur magna.

Fala Professor

3.1 INSTRUÇÕES INICIAIS PARA UTILIZAR O PHP

O PHP teve seu início em meados do ano de 1994, sendo que sofreu diversas modificações ao longo dos anos até que pudesse chegar à linguagem que conhecemos atualmente (PHP versão 5).

Uma das grandes marcas do PHP é em relação ao suporte. Vejamos:

- **Plataformas suportadas**

Está disponível para várias plataformas, entre elas Windows, Linux, FreeBSD, Mac OS etc.

- **Banco de dados suportados**

Suporta diversos bancos de dados, sendo que o MySQL já vem como banco de dados nativo (principal). Mas não se preocupe, você pode ativar os outros banco de dados de maneira bem simples. Falaremos disso em seguida, no capítulo 3.2. Entre os bancos com suporte podemos citar PostgreSQL, Interbase, Firebird, Oracle, Sybase, MySQL, SQL Server (MSSQL) etc.

Características gerais do PHP:

Distribuído sob a licença GPL, possui seu código-fonte (código utilizado para sua criação) aberto, o que facilita a correção de eventuais erros no código, permitindo seu rápido desenvolvimento.

Para que o PHP funcione, necessitamos que se tenha instalado no servidor:

- PHP (com suas bibliotecas);
- o servidor web responsável por interpretar a linguagem PHP e convertê-la para HTML. Normalmente se usa o Apache (Windows, Linux etc.) ou o IIS - *Internet Information Services* (Windows).
- um banco de dados (caso você deseje utilizar um).

Atenção



É importante salientar que esses programas serão instalados somente no servidor, ou seja, quem acessar o site não vai precisar ter esses programas instalados no seu computador. Um exemplo claro disso é quando você acessa o site da Globo.com. Esse site é dinâmico e utiliza uma linguagem *server-side*. No entanto, você não precisa instalar a linguagem serve-side, o servidor web e nem o banco de dados em sua máquina para acessar este site.

Existem vários utilitários (pacotes) que já instalam os programas necessários para a execução do PHP, e ainda vêm com alguns requintes a mais, tais como banco de dados e o PHPMyAdmin (software web que permite administrar o banco de dados). Na tabela abaixo, irei citar alguns:

| Programa | Link |
|-------------------|---|
| EasyPHP | http://www.easyphp.org |
| PHPTriad | http://www.phpriad.org |
| WampServer | http://www.wampserver.com |
| Xampp | http://www.apachefriends.org/pt_br/xampp.html |

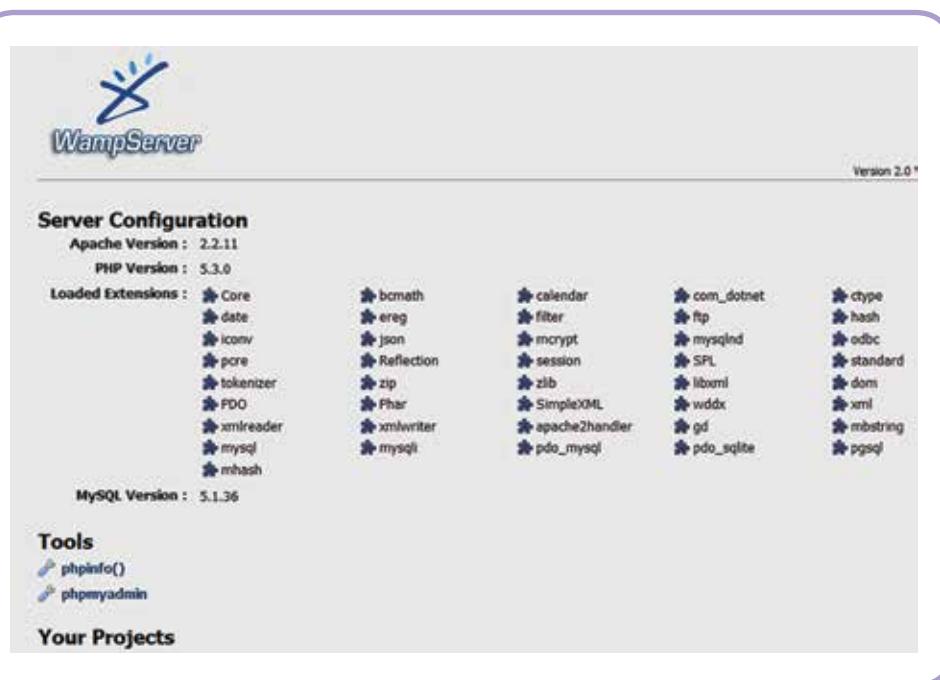
Independente do utilitário que você escolher, você deve optar sempre por baixar a versão mais recente dele, pois assim, tanto o PHP quanto o MySQL estarão em suas versões mais atualizadas.

Então, agora vamos botar a mão na massa. Para meus exemplos, irei considerar a utilização do WampServer.

Uma vez instalado, necessitamos verificar se o PHP está funcionando. Para isso, siga os passos abaixo:

- abra o navegador (browser);
- na barra de endereços, digite `http://localhost` ou `http://127.0.0.1`
- Se tudo correu bem, você deverá se deparar com uma tela parecida com a da figura 29:

Figura 29 - Tela inicial do Wamp Server



Perceba que aparecem diversas informações na tela. Em *Server Configuration* aparecem as configurações do servidor. Repare que neste item são exibidas as versões e as extensões “ativas” no PHP. Mais abaixo aparecem as *Tools* (ferramentas) disponíveis no PHP. O PHPInfo() é a que nos permite efetuar as devidas configurações no PHP (veremos no capítulo 3.2); já o PHPMyAdmin é a ferramenta que nos permite administrar o banco de dados (veremos no capítulo 4).

Caso não apareça a tela como mostrada na figura 29, verifique se digitou o caminho corretos na barra de endereços.



Atenção

Atenção

Ou, ainda, um erro muito comum nesta etapa é não iniciar o servidor Web. Para isso, verifique junto à área dos ícones de notificação (ao lado do relógio) se o servidor web está ativo. Caso não esteja, basta clicar no ícone e clicar novamente em “Start All Services” (iniciar todos os serviços).

3.2 CONFIGURANDO O PHP

Para acessar e alterar as configurações do PHP, basta acessar o arquivo php.ini. Para saber onde este arquivo está localizado na sua máquina, acesse o `phpinfo()` conforme explicamos no capítulo anterior. Você verá uma tela parecida como esta abaixo:

Figura 30 - Tela do `phpinfo()`

| | |
|---|---|
| | oci8=D:\php-sdk\oracle\instantclient10\ sdk\shared" --with-oci8-11g=D:\php-sdk\oracle\instantclient11\ sdk\shared" --with-enchant=shared" |
| Server API | Apache 2.0 Handler |
| Virtual Directory Support | enabled |
| Configuration File (php.ini) Path | C:\Windows |
| Loaded Configuration File | C:\wamp\bin\apache\Apache2.2.11\bin\php.ini |
| Scan this dir for additional .ini files | (none) |
| Additional .ini files parsed | (none) |
| PHP API | 20090626 |

Observe que a linha “*Loaded Configuration File*” é onde se encontra o arquivo php.ini que vamos utilizar para efetuar as configurações do PHP.

Utilize um editor de texto para abrir o arquivo php.ini. Recomendo utilizar o **NotePad** (Bloco de Notas) no Windows ou o vi no Linux, por exemplo.

Uma vez aberto o php.ini, basta procurar a configuração desejada e efetuar as devidas alterações. É importante destacar que, para habilitar alguma dll (um banco de dados, por exemplo), é necessário retirar o “;” (ponto e vírgula) que fica antes do nome da dll.

A seguir, listamos algumas configurações que merecem nossa atenção:

| Configuração | Descrição |
|------------------------------|--|
| Limites de Recursos | |
| | Integer |
| max_execution_time | Define o limite de tempo de execução de um script antes que seja terminado pelo interpretador. Isto ajuda a prevenir que scripts mal escritos serem executados indefinidamente pelo servidor. O padrão é 30. |
| | Integer |
| memory_limit | Configura a quantidade máxima de memória (bytes) que o script pode alocar. Isso ajuda a prevenir que scripts mal feitos consumam toda a memória disponível em um servidor. Utilize -1 se você não quiser impor um limite de memória. |
| Manuseio de dados | |
| | String |
| default_charset | Tipo de codificação de caracteres no cabeçalho "Content-type:" Para desabilitar o envio de um conjunto de caracteres, simplesmente deixe essa diretiva vazia. (Retire o "," para habilitá-la) |
| | String |
| default_mimetype | Tipo de conteúdo no cabeçalho "Content-type:" |
| | Integer |
| post_max_size | Informa o máximo de tamanho permitido no método POST. Esta configuração afeta também o upload de arquivos. |
| Caminhos e diretórios | |
| extension_dir | Local onde estão armazenadas as bibliotecas (dll) que devem ser carregadas. |
| Envio de arquivos | |
| | Boolean |
| file_uploads | Se deve ou não permitir envio de arquivos via HTTP. |
| | String |
| upload_tmp_dir | O diretório temporário usado para guardar arquivos quando arquivos forem enviados. O usuário do PHP deve ter permissão de escrita nesse diretório. Se não for especificado, o PHP usará o padrão do sistema. |
| | Integer |
| upload_max_filesize | O tamanho máximo de um arquivo enviado. |
| Opções Gerais de SQL | |
| | Boolean |
| sql.safe_mode | Se ativada, funções de conexão com banco de dados que especificam valores padrão usarão esses valores ao invés dos argumentos fornecidos. |
| Extensões dinâmicas | |
| php_gd2.dll | Quando habilitada, esta biblioteca permite a execução de funções gráficas no PHP. |
| php_mbstring.dll | Quando habilitada, esta biblioteca permite o uso de caracteres <i>multibyte</i> . Muito utilizada em logins de acesso. |

| Configuração | Descrição |
|--------------------------------|--|
| Extensões dinâmicas | |
| php_mysql.dll | Quando habilitada, esta biblioteca permite a conexão com o banco de dados MySQL. |
| php_mssql.dll | Quando habilitada, esta biblioteca permite a conexão com o banco de dados SQL Server. |
| php_pgsql.dll | Quando habilitada, esta biblioteca permite a conexão com o banco de dados PostgreSQL. OBS.: É necessário, ainda, copiar a dll libpq.dll da pasta do PHP para a pasta BIN do Apache. |
| Configurações de Módulo | |
| date.timezone | <p>String</p> <p>Altera a região de onde será puxada a data e hora. Veja mais em http://php.net/date.timezone</p> <p>OBS.: Caso queria pegar o horário de uma outra região, opcionalmente podemos utilizar o código abaixo no nosso script: <code>setlocale (LC_ALL,'pt_BR','ptb','portuguese-brazil','bra','brazil');</code> echo strftime ("%H:%M");</p> |
| Funções de E-mails | |
| SMTP | <p>String</p> <p>Especifica o seu servidor de email. Ex.: SMTP = smtp.flavioizo.com</p> |
| SMTP_PORT | <p>Integer</p> <p>Especifica a porta pela qual utiliza o servidor de e-mail.</p> |
| sendmail_from ou sendmail_path | <p>String</p> <p>Especifica o seu e-mail (e-mail de saída). O sendmail_from é para quem utiliza Windows e sendmail_path para quem utiliza o Linux.</p> |
| Configurações - MySQL | |
| mysql.default_host | <p>String</p> <p>Define o host padrão para a função mysql_connect().</p> |
| mysql.default_user | <p>String</p> <p>Define o usuário padrão para a função mysql_connect().</p> |
| mysql.default_password | <p>String</p> <p>Define a senha padrão para a função mysql_connect(). OBS.: Não é recomendado definir a senha, a não ser para testes, pois qualquer um pode ter acesso ao arquivo php.ini e ver qual a senha do BD. Ou, ainda, pode ser feito um <code>echo get_cfg_var("mysql.default_password")</code> e a senha será impressa.</p> |
| mysql.connect_timeout | <p>Integer</p> <p>Define o máximo de tempo (em segundos) que a conexão com o banco ficará aberta. O padrão é 60. Coloque "-1" para definir sem limites.</p> |
| Configurações de Sessão | |
| session.save_path | <p>String</p> <p>Define o local onde os arquivos serão criados.</p> |

| Configuração | Descrição |
|--------------------------------|---|
| Configurações de Sessão | |
| session.name | String Especifica o nome da sessão. |
| session.auto_start | Integer Define se o módulo da sessão inicia automaticamente em todas as páginas. O padrão é 0 (desabilitado), sendo assim, deve-se utilizar a função session_start() nas páginas em que se deseja iniciar o módulo de sessão. |
| session.gc_maxlifetime | Integer Especifica o número de segundos, após os dados terem sido considerados como lixo, ("garbage") para serem limpos. |
| session.cache_expire | Integer Define o número de minutos para a sessão expirar. O valor padrão é 180. |
| session.hash_function | Integer Define o tipo de hash: 0 para "MD5 128 bits" e 1 para "SHA-1 160 bits". Disponível a partir do PHP 5 |

Essas são as configurações básicas do php.ini. No entanto, existem outras, que você pode ir descobrindo ao longo de suas práticas com a linguagem. Acesse o arquivo php.ini (conforme visto na figura 30) e conheça-o melhor.

Uma situação muito comum é efetuar as alterações no php.ini e estas não serem devidamente aplicadas. Sendo assim, não esqueça que, após fazer as alterações no arquivo php.ini, é necessário salvá-lo e reiniciar o servidor WEB.



Atenção

3.3 UTILIZAÇÃO DE FERRAMENTAS PARA DESENVOLVIMENTO WEB

Existem diversas ferramentas que podem colaborar com o desenvolvimento Web. Algumas são ferramentas proprietárias (pagas) e outras são gratuitas. Façamos uma classificação com as principais.

| Ferramenta | Descrição |
|-------------------|---|
| Adobe Dreamweaver | Possibilita a criação e o gerenciamento de páginas Web. http://www.adobe.com/br/products/dreamweaver.html Software Proprietário |

| Ferramenta | Descrição |
|------------------------|---|
| Adobe Photoshop | Possibilita a criação de conteúdos gráficos. http://www.adobe.com/br/products/photoshop.html Software Proprietário |
| Adobe Flash | Possibilita a criação de conteúdo interativo. http://www.adobe.com/br/products/flash.html Software Proprietário |
| Notepad++ | Editor de texto e códigos fonte que suporta as mais variadas linguagens de programação. Recurso mais avançado que o bloco de notas (notepad). http://notepad-plus-plus.org Software Gratuito |
| NuSphere PHPED | Editor de conteúdos PHP http://www.phped.com Software Proprietário |
| PHP Editor | Editor de conteúdos PHP http://www.baixaki.com.br/download/php-editor.htm Software Gratuito |
| PHP Designer | Editor de conteúdos PHP http://www.mpsoftware.dk/phpdesigner.php Software Proprietário |
| NetBeans | Editor para desenvolvimento desktop, mobile e para aplicação Web. Suporta Java, PHP, C/C++ entre outros. http://www.netbeans.org Software Gratuito |
| Zend Studio | Editor para desenvolvimento de aplicações Web baseadas em PHP. http://www zend com products studio Software Proprietário |

Essas são apenas algumas ferramentas, existem outras que você também poderá utilizar para desenvolver seu código PHP. No entanto, é importante frisar que até com um editor bem simples como o notepad é possível criar um site completo. O que as ferramentas, citadas acima, possuem é um diferencial para facilitar o nosso trabalho.

Lembre-se também de que a ferramenta sozinha não irá valer de nada se você não souber manipulá-la e nem entender a linguagem por meio da qual vai desenvolvê-la. Isso, pois as ferramentas só auxiliam nossos afazeres.

3.4 INTRODUÇÃO À SINTAXE BÁSICA

Em se tratando de sintaxe, o PHP é muito parecido com C e C++.

3.4.1 Imprimindo conteúdo na tela:

Figura 31 - Código PHP + HTML

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Primeiro site</title>
  </head>
  <body>
    <?php
      // Abaixo será impresso a mensagem "Olá Mundo"
      echo "<h2>Olá Mundo</h2>";
    ?>
  </body>
</html>
```

De acordo com a figura acima, vamos detalhar o que se vê:

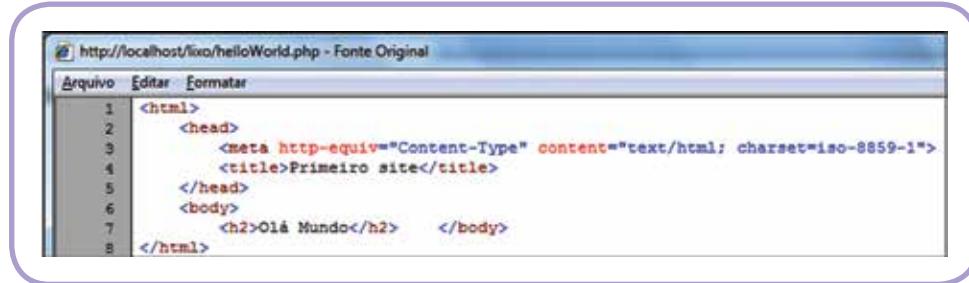
- `<?php` inicia o código PHP.
- `?>` finaliza o código PHP.
- `// ou #` é o início do comentário de uma linha no PHP. Para comentar blocos de linhas, utilize `/*` linhas com os comentários `*/`
- `echo` é o comando de impressão do PHP. Equivale ao “printf” do C, ou ao “write” do javascript.
- Outro ponto a ser observado é que dentro do “echo” nós podemos fazer a impressão de códigos HTML.

Tudo o que for colocado fora das tags `<?php ?>` será considerado HTML. Os blocos de código PHP podem ser abertos e fechados diversas vezes dentro da mesma página PHP.

É importante observar que, ao final de cada comando, devemos colocar um ponto e vírgula (`;`).

O código PHP fica embutido no código HTML. Sendo assim, conforme vimos na figura 03, quando efetuamos a solicitação de interpretação de uma linguagem, o servidor WEB devolve somente códigos HTML, imagens etc., ou seja, o código *Server-Side* é interpretado e transformado em HTML e devolvido para o browser. Isso fica claro se, ao executar o código da figura 31, pedirmos para visualizar o código fonte. Olha o que teremos de retorno:

Figura 32 - Código fonte da figura 31 visto no Internet Explorer



A screenshot of a web browser window titled "http://localhost/lixo/helloWorld.php - Fonte Original". The menu bar includes "Arquivo", "Editar", and "Formatar". The code editor shows the following PHP code:

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Primeiro site</title>
5   </head>
6   <body>
7     <h2>Olá Mundo</h2>
8   </body>
```

Veja que a sintaxe PHP não aparece, somente visualizamos a interpretação que o servidor Web fez do código PHP. Esse procedimento evita que o usuário copie seu código PHP, visualize suas senhas de conexão com o banco de dados etc.

Atenção

Lembre-se: toda página em que for utilizar código PHP deverá ser salva com a extensão **.php**

Assim, páginas que só terão código HTML e/ou javascript poderão ser salvas como **.htm** ou **.html**

No entanto, páginas que contenham código HTML e/ou Javascript e PHP **OBRIGATORIAMENTE** deverão ser salvas com extensão **.php**

3.5 TIPOS DE DADOS

O PHP é uma linguagem não tipada (variáveis com valores dinâmicos), ou seja, não há a necessidade de declarar o tipo da variável. Assim, durante a execução do script uma variável pode receber valores do tipo “String” (texto) e, posteriormente, receber valores do tipo “integer” (inteiro).

Pode-se, ainda, “forçar” uma variável a receber um tipo determinado de dados. Para isso, coloque o tipo de dados entre “parênteses” antes de atribuir o valor à variável. Exemplo:

```
$salario = (string)900.00;
```

No exemplo acima, o valor *double* 900.00 está sendo forçado a virar um valor do tipo string, ou seja, o mesmo que “900.00”.

No tópico a seguir, veremos mais sobre variáveis.

3.6 VARIÁVEIS

Em PHP, não é necessário declarar as variáveis, basta que o programador atribua um valor à variável e esta já estará guardada na memória. Para criar variáveis em PHP é necessário seguir duas regras:

- as variáveis em PHP obrigatoriamente devem começar com o caractere especial cifrão (\$);
- as variáveis são case sensitive, ou seja, diferem letras maiúsculas de minúsculas.

Vejamos alguns exemplos: \$aluno, \$professor, \$dataDeNascimento.

Abaixo temos um exemplo aplicado:

Figura 33 - Script em PHP - Variáveis

```

1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Trabalhando com variáveis</title>
5   </head>
6   <body>
7     <?php
8       $x = 10;
9       $y = 8;
10      $total = $x + $y;
11      echo "A soma de $x com $y é igual a $total";
12      echo "<br />";
13      echo "A soma de " . $x . " com " . $y . " é igual a " . $total;
14    ?>
15   </body>
16 </html>
```

Observe que as variáveis não foram declaradas. Vejam também que tanto a linha 11 quanto a linha 13 imprimem o mesmo resultado. Na linha 13, especificamente, eu utilizei um recurso denominado concatenação (ver tópico 3.6). Esse recurso junta vários trechos e imprime-os com a utilização de um único “echo”. Para juntar esses valores, eu utilizei o caractere ponto (.).

Figura 34 - Resultado da interpretação da figura 33



A soma de 10 com 8 é igual a 18
A soma de 10 com 8 é igual a 18

Conforme se pode perceber, além de não declarar, também não informamos o tipo de dados das variáveis (veja o tópico 3.5), pois isso não é necessário no PHP.

Já quando utilizamos variáveis do tipo Array, a situação muda um pouco. Esse tipo de variável pode ser montado de diversas formas:

Forma 01:

```
$lista = array(10,29,34,6,25);
```

Esse *array* pode ser chamado tendo como auxílio o caractere especial `[]`. Vejamos:

```
echo $lista[3]; // vai imprimir o valor 6  
echo $lista[1]; // vai imprimir o valor 29
```

OBS.: O *array* começa na posição 0 (zero).

Forma 02:

```
$lista[0] = 10; //valor 10 na posição zero;  
$lista[1] = 11; //valor 11 na posição 1;  
$lista[] = 12; //valor 12 na posição 2;  
$lista[5] = 15; //valor 14 colocado na posição 5;
```

Esse código vai gerar um *array* da seguinte forma:

```
$lista = array(10, 11, 12, ‘’, ‘’, 15);
```

Forma 03:

Os *arrays* também podem conter strings ao invés de números:

```
$usuario[‘nome’] = “Flávio Izo”;  
$usuario[‘idade’] = 29;  
$usuario[‘email’] = “flavio@flavioizo.com”;
```

Esse *array* é o mesmo que nós fizéssemos assim:

```
$usuario = array(“nome” => “Flávio Izo”, “idade” => 29, “email” =>  
“flavio@flavioizo.com”);
```

Agora podemos efetuar a impressão dos valores:

```
echo "O usuário " . $usuario['nome'] . ", de " . $usuario['idade'] . " anos
      tem email " . $usuario['email'] . ":";
```

Forma 04:

Há ainda as matrizes bidimensionais:

```
$populacao['ES']['Cachoeiro'] = "250.000";
$populacao['ES']['Vitória'] = "300.000";
$populacao['BA']['Salvador'] = "350.000";
$populacao['MG']['Juiz de Fora'] = "500.000";
```

Para saber mais acerca das variáveis, pesquise sobre:

- Modificador static.



Indicações

Variáveis Globais e Locais:

Toda variável possui um escopo, podendo este ser global ou local. O escopo é o bloco de comandos em que a variável pode ser utilizada. Na maioria das vezes, uma variável é global, ou seja, pode ser utilizada em qualquer parte do código. Já a local, só é utilizada dentro do seu bloco de criação.

Quando elaboramos um *script* que contenha funções, só podemos utilizar as variáveis declaradas dentro da própria função. Uma variável global **não** pode ser utilizada dentro de uma função sem que seja “declarada” como global. Exemplo:

Figura 35 - Modo incorreto de utilizar variável global

```

1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Variáveis</title>
5   </head>
6   <body>
7     <?php
8       $frase = "O rato roeu a roupa do rei de Roma!";
9       function imprime() {
10         echo $frase;
11       }
12       echo "Página WEB <br />";
13       echo "Vamos chamar agora a função imprime() <br />";
14       imprime();
15     ?>
16   </body>
17 </html>
```

Se executarmos esse código, vamos obter o seguinte erro:

Notice: Undefined variable: frase in C:\wamp\www\testes\globaisLocais01PHP.php on line 10

Como resolver isso? Veja:

Figura 36 - Modo correto de utilizar variável global

```
1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Variáveis</title>
5   </head>
6   <body>
7     <?php
8       $frase = "O rato roeu a roupa do rei de Roma!";
9       function imprime(){
10         global $frase;
11         echo $frase;
12       }
13       echo "Página WEB <br />";
14       echo "Vamos chamar agora a função imprime() <br />";
15       imprime();
16     ?>
17   </body>
18 </html>
```

Existe ainda outra forma de acessar variáveis globais. Veja o exemplo anterior, porém utilizando o *array superglobal* \$GLOBALS:

Figura 37 - Variável global utilizando o \$GLOBALS

```
1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Variáveis</title>
5   </head>
6   <body>
7     <?php
8       $frase = "O rato roeu a roupa do rei de Roma!";
9       function imprime(){
10         echo $GLOBALS['frase'];
11       }
12       echo "Página WEB <br />";
13       echo "Vamos chamar agora a função imprime() <br />";
14       imprime();
15     ?>
16   </body>
17 </html>
```

Então, sempre que necessitar, utilize uma das duas formas para recuperar uma variável global.

3.7 CONSTANTES

As constantes (identificadores que, ao receber um valor, não pode mais ser alterados até o fim da execução do script) são estabelecidas utilizando-se a função `define("nome_da_constante",valor)`. E para recuperar um valor de uma constante, você deve utilizar os mesmos procedimentos que utiliza para chamar uma variável, porém sem o \$ (cifrão). Outra alternativa para chamar uma constante é utilizar a função `constant("nome_da_constante")`.

Veja o exemplo:

Figura 38 - Utilizando constantes

```

1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Trabalhando com variáveis</title>
5   </head>
6   <body>
7     <?php
8       define("x",10);
9       define("y",8);
10      $total = constant('x') + constant('y');
11      echo "A soma de ".x." com ".y." é igual a ".$total;
12      define("x",15);
13    ?>
14   </body>
15 </html>

```

Veja que, na figura 35, nas linhas 8 e 9 eu defini duas CONSTANTES (x e y). Na linha 10, eu recuperei esses valores utilizando a função `contant()`. Já na linha 11, eu recuperei os valores das constantes x e y chamando diretamente por x e y (sem o uso da função `constant()`).

É importante observar, ainda, que esse código possui um erro no escopo de montagem. Você é capaz de descobrir em qual linha está o erro?

Se você respondeu que está na linha 12, você acertou! A constante não pode ter seu valor alterado durante a execução do script. Se eu executar essa página, olha o que vai acontecer:

Figura 39 - Executando a página “constantes.php”

A soma de 10 com 8 é igual a 18
Notice: Constant x already defined in C:\wamp\www\testes\constantes.php on line 12

Nós recebemos uma mensagem (*notice*) informando que uma constante x já foi definida!

O PHP possui algumas constantes que já vêm pré-definidas. Essas constantes nos auxiliam para que recebamos algumas informações úteis. Relacionei algumas:

| Constante | Descrição |
|--------------------------|---|
| PHP_VERSION | Mostra a versão do PHP. |
| PHP_MAJOR_VERSION | Mostra a versão principal do PHP. |
| PHP_MINOR_VERSION | Mostra a versão secundária do PHP. |
| PHP_OS | Mostra o sistema operacional onde está instalado o PHP. |
| PHP_EXTENSION_DIR | Mostra o diretório onde ficam as extensões. |

Para visualizar todas as constantes existentes na sua versão PHP, utilize o código abaixo:

```
echo '<pre>';
print_r(get_defined_constants());
echo '</pre>';
```

A função `get_defined_constants()` retorna um “array” com todas as constantes que estão atualmente disponíveis, inclusive as que foram definidas por você. Já a função `print_r()` imprime todo o conteúdo do “array”.

Vale lembrar que, conforme visto no tópico 2.6, a tag `<pre>` formata o texto conforme foi escrito.

Atenção



Utilize a função `defined("nome_da_constante")` para descobrir se uma constante existe (foi declarada).

Quando executada, esta função retornará `1` (ou `true`) para verdadeiro ou `0` (ou `false`) para falso.

Exemplo:

```
if (defined("x") == true) {
    echo "a constante x existe!";
} else {
    echo "a constante x não existe!";
}
```

3.8 CONCATENANDO VALORES

A concatenação de valores é uma técnica que auxilia muito as linguagens de programação. No PHP utilizamos o ponto (.) para concatenar os valores. Veja:

Figura 40 - Concatenando textos

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Primeiro site</title>
  </head>
  <body>
    <?php
      //Abaixo será impresso a mensagem "Olá Mundo"
      echo "<h2>Olá Mundo!</h2> ." "Sou novo por aqui." ;
    ?>
  </body>
</html>
```

O mesmo vale para a concatenação de variáveis. Veja que na figura 35 nós fizemos exatamente isso: a concatenação de variáveis e textos (*strings*).

Agora teste seus conhecimentos. O que será impresso em cada um dos três códigos abaixo?

```
$nome = "Izo";
echo "Meu nome é ";
echo $nome;
```

```
$nome = "Izo";
echo "Meu nome é $nome";
```

```
$nome = "Izo";
echo 'Meu nome é ' . $nome;
```

Resposta: A mesma coisa, ou seja, “Meu nome é Izo”.

3.9 ARRAYS SUPER GLOBAIS

A linguagem PHP nos fornece alguns *arrays* nativos (built-in), ou seja, são palavras reservadas que já existem e estão prontas para serem utilizados. Eu diria até que essa é uma das partes mais importantes do PHP. Veja quais são eles:

- Os arrays `$_GET` e `$_POST` servem para receber valores vindos dos formulários, de acordo com o método definido.

- O `$GLOBALS` serve para receber valores de variáveis globais, conforme mostrado no tópico 3.6 Variáveis.
- O `$_SESSION` é utilizado sempre que trabalhamos com sessões (veremos mais à frente, no tópico “Sessões”).
- O `$_COOKIE` utilizado quando trabalhamos com cookies (arquivos de rastreamento salvos no computador do cliente).

3.10 OPERADORES

Os operadores são muito importantes dentro de projetos dinâmicos. Vamos conhecê-los:

3.10.1 Operadores Aritméticos av:

| Operadores | Matemática | Javascript |
|----------------------|--------------|------------|
| Adição | $A + B$ | $A + B$ |
| Subtração | $A - B$ | $A - B$ |
| Multiplicação | $A \times B$ | $A * B$ |
| Divisão | $A \div B$ | A / B |
| Módulo (mod) | | $A \% B$ |

Para achar o DIV, utilize a função nativa de PHP `floor()` que arredonda o resultado para baixo.

É possível atribuir uma operação a uma variável em tempo de execução. Como?

- `$variavel = $x + $y;`
- `$variavel = $x + $y * $z;`
- `$variavel = $x / $y + $z;`
- `$variavel = ((($x + $y) * $z) / $p);`

Operadores relacionais (de comparação):

Esses tipos de operadores nos auxiliam na resolução de cálculos envolvendo expressões relacionais. Devemos lembrar que os operandos da relação devem ser do mesmo tipo e o resultado sempre será lógico (verdadeiro ou falso). Vejamos:

| Operadores | Matemática | PHP |
|---------------------|------------|--|
| Igual a | $A = B$ | <code>\$A == \$B</code> |
| Diferente de | $A \neq B$ | <code>\$A != \$B</code> <code>\$A <> \$B</code> |

| Operadores | Matemática | PHP |
|-------------------------|------------|----------------|
| Maior que | $A > B$ | $\$A > \B |
| Menor que | $A < B$ | $\$A < \B |
| Maior ou igual a | $A \geq B$ | $\$A \geq \B |
| Menor ou igual a | $A \leq B$ | $\$A \leq \B |

Operadores de atribuição:

Esses tipos de operadores nos auxiliam na atribuição de valores às variáveis. Vejamos:

| Operadores | Significado | PHP |
|------------|----------------|---------------|
| $+=$ | $A = A + B$ | $\$A += \B |
| $-=$ | $A = A - B$ | $\$A -= \B |
| $*=$ | $A = A * B$ | $\$A *= \B |
| $/=$ | $A = A \div B$ | $\$A /= \B |
| $%=$ | $A = A \% B$ | $\$A \%=\ B |
| $++$ | $A = A + 1$ | $\$A++$ |
| $--$ | $A = A - 1$ | $\$A--$ |

Operadores lógicos:

Esses tipos de operadores nos auxiliam nos testes lógicos. Vejamos:

| Operadores | Significado | PHP, Onde A=9 e B=4 |
|--------------------------|--------------|---|
| ! ou not | NÃO | $!(\$A = 5)$ obtém-se "1" |
| or ou | OU | $(\$A < 5) \text{ or } (\$B > 3)$ obtém-se "1" |
| and ou && | E | $(\$A < 5) \text{ and } (\$B > 3)$ obtém-se "0" |
| xor | OU EXCLUSIVO | $(\$A > 5) \text{ xor } (\$B > 2)$ obtém-se "0" |

Operador Ternário:

Esse tipo de operador tem um entendimento especial. Ele atribui um valor entre dois possíveis, de acordo com o resultado da condição. Sua sintaxe é bem simples:

`(Condição) ? (Valor1) : (Valor2);`

Observe o uso dos sinais `?` e `:` para testar a condição. Se ela retornar um valor verdadeiro (*true*), *Valor1* será executado. Caso o retorno seja negativo (*false*), teremos a execução de *Valor2*.

`$c = ($a == $b) ? ($a - $b) : ($a + $b);`

3.11 ESTRUTURAS CONDICIONAIS

Já vimos no javascript que as estruturas condicionais são importantes em linguagens de programação. No PHP não é diferente.

Existem dois tipos de estruturas condicionais: if e switch case.

If

É a estrutura mais utilizada. Verifica qualquer tipo de condição. Veja como podemos montar sua sintaxe:

Figura 41 - Estrutura condicional IF

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Estrutura condicional em PHP</title>
5   </head>
6   <body>
7     <?php
8       $salario = 8000.00;
9       $dividas = 6580.00;
10      if ($salario > $divida){
11        echo "Você terá saldo positivo";
12      }else{
13        echo "Você terá saldo negativo";
14      }
15    ?>
16
17   </body>
18 </html>
```

Switch Case

É a estrutura que verifica condições que são mais específicas. Veja sua sintaxe:

Figura 42 - Estrutura condicional Switch Case

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Estrutura de repetição em PHP</title>
5   </head>
6   <body>
7     <?php
8       $cor = "azul";
9       switch ($cor){
10         case "vermelha":
11           echo "A cor escolhida é vermelha <br />";
12           echo "Essa cor representa paixão, energia";
13           break;
14         case "azul":
15           echo "A cor escolhida é azul <br />";
16           echo "Essa cor representa tranquilidade, harmonia, serenidade";
17           break;
18         case "branca":
19           echo "A cor escolhida é branca <br />";
20           echo "Essa cor representa paz";
21           break;
22         default:
23           echo "Essa cor não existe!";
24       }
25     ?>
26   </body>
27 </html>
```

3.12 ESTRUTURAS DE REPETIÇÃO

As estruturas de repetição existentes no PHP são: *while*, *do while*, *for* e *foreach*. Veja cada uma delas:

While

A estrutura *while* analisa uma expressão e executa o bloco de comandos específico enquanto a expressão for verdadeira.

Figura 43 - Estrutura de repetição “*while*”

```

1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Estrutura de repetição em PHP</title>
5   </head>
6   <body>
7     <?php
8       $cont = 0;
9       while ($cont < 10) {
10         echo "Valor de \$cont: $cont <br />";
11         $cont++;
12       }
13     ?>
14   </body>
15 </html>
```

Obs.: Sempre deve haver pelo menos uma chance de a expressão ser falsa.

Do While

Muito parecida com a estrutura *while*, porém esta executa o bloco de comandos para depois analisar a expressão.

Figura 44 - Estrutura de repetição “*do while*”

```

1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Estrutura de repetição em PHP</title>
5   </head>
6   <body>
7     <?php
8       $cont = 0;
9       do {
10         echo "Valor de \$cont: $cont <br />";
11         $cont++;
12       }while ($cont < 10)
13     ?>
14   </body>
15 </html>
```

For

É uma estrutura de repetição como as outras, porém, na sua própria sintaxe são definidas algumas regras:

- Inicialização: o valor de início do contador. Ex.: \$x=1
- Condição: a condição para que a repetição termine.
- Operador: o incremento ou decremento no valor do contador de inicialização. Ex.: \$x++

Figura 45 - Estrutura de repetição “for”

```
1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Estrutura de repetição em PHP</title>
5   </head>
6   <body>
7     <?php
8       for($cont=0; $cont<10; $cont++){
9         echo "Valor de \$cont: $cont <br />";
10      }
11    ?>
12   </body>
13 </html>
```

Foreach

Esta estrutura oferece uma maneira mais simples de percorrer os elementos de um array. Pode ser montado de duas formas. Vejamos:

Figura 46 - Estrutura de repetição “foreach”. Exemplo 1

```
1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4     <title>Estrutura de repetição em PHP</title>
5   </head>
6   <body>
7     <?php
8       $valor = array(1, 2, 3, 4);
9       foreach ($valor as $x){
10         echo "O valor é: $x <br />";
11       }
12     ?>
13   </body>
14 </html>
```

A figura 46 gera a seguinte saída:

O valor é: 1

O valor é: 2

O valor é: 3

O valor é: 4

Figura 47 - Estrutura de repetição “foreach”. Exemplo 2

```

1  <html>
2    <head>
3      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4      <title>Estrutura de repetição em PHP</title>
5    </head>
6    <body>
7      <?php
8      $lista = array("nome" => "Flávio", "Idade" => "29", "Cidade" => "Cachoeiro");
9      foreach ($lista as $x => $valor){
10        echo "\$lista[$x] aponta para: $valor <br />";
11      }
12    ?>
13    </body>
14  </html>

```

A figura 47 gera a seguinte saída:

```

$listado[nome] aponta para: Flávio
$listado[Idade] aponta para: 29
$listado[Cidade] aponta para: Cachoeiro

```

Agora que você já conheceu os itens que fazem parte da estrutura de repetição, quando necessitar utilizá-la, escolha sempre a mais adequada para o seu problema.

3.13 FUNÇÕES

As funções são blocos de comandos com o objetivo de executar determinada ação. A sintaxe básica da função é:

```

<?php
function nome_funcao(arg_1, arg_2, arg_n){
    comandos;
}
?>

```

É importante saber que as funções podem ou não retornar valores. No caso de não retornar valor, a função somente executa um ou vários comandos e pronto! Já no caso de retornar valor, este pode ser uma variável contendo um valor único ou um array.

As funções também podem receber parâmetros, sendo esses separados por vírgula e passados durante a chamada da função.

Vejamos alguns exemplos:

Exemplo 01:

Figura 48 - Exemplo de uso de funções

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Funções em PHP</title>
5   </head>
6   <body>
7     <?php
8       function mensagem(){
9         echo "Escreva uma mensagem. <br />";
10        echo "Esse é um exemplo de função <br />";
11        for ($i=1;$i<=3;$i++){
12          echo "Utilizando for na função <br />";
13          echo "valor de \$i: $i <br />";
14        }
15      }
16      echo "Página WEB <br />";
17      echo "Vamos chamar agora a função mensagem() <br />";
18      mensagem();
19      echo "<br /> A função foi executada!";
20    ?>
21   </body>
22 </html>
```

Exemplo 2:

Como já falamos, uma função poderá dar retorno ou não. Em caso de retornar algum valor, é necessário colocar o “return” ao fim da execução da função.

Figura 49 - Exemplo de uso de funções

```
1 <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Funções em PHP</title>
5   </head>
6   <body>
7     <?php
8       function retornaValor(){
9         return 100;
10      }
11      echo "Página WEB <br />";
12      $valor = 0;
13      echo "A variável \$valor possui valor $valor <br />";
14      echo "Vamos chamar agora a função retornaValor() <br />";
15      $valor = retornaValor();
16      echo "A variável \$valor possui valor $valor <br />";
17    ?>
18   </body>
19 </html>
```

Toda vez que chamamos uma função que retorna um valor, este deverá ser OBRIGATORIAMENTE atribuído a alguma variável, impresso na

tela ou utilizado em um cálculo. No exemplo acima, eu atribuí o retorno da função à variável \$valor.

Exemplo 3:

Pode-se, ainda, chamar funções passando-se alguns parâmetros (argumentos). Veja:

Figura 50 - Exemplo de uso de funções

```

1  <html>
2   <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4     <title>Funções em PHP</title>
5   </head>
6   <body>
7     <?php
8       function imprime($parametro) {
9         echo $parametro;
10      }
11      echo "Página WEB <br />";
12      $texto = "Estou aprendendo a utilizar função! <br />";
13      echo "Vamos chamar agora a função imprime() <br />";
14      imprime($texto);
15    ?>
16   </body>
17 </html>
```

Funções Built-in (internas) do PHP:

Existem algumas funções que são internas do PHP e podem nos auxiliar muito na execução de scripts:

| Função | Descrição |
|---------------------------|--|
| htmlspecialchars() | Converte caracteres especiais para a realidade HTML |
| addslashes() | Retorna uma string com barras invertidas antes de caracteres que precisam ser escapados (para serem utilizados em query a banco de dados, por exemplo). Estes caracteres são aspas simples ('), aspas duplas ("), barra invertida (\) e NUL (o byte NULL). |
| stripslashes() | Desfaz o efeito de addslashes(), ou seja, remove barras invertidas de uma string. |
| urlencode() | Retorna uma string em que todos os caracteres não-alfanuméricos com exceção de _ são substituídos com um sinal de porcento (%) seguido por dois dígitos hexadecimais. Os espaços são codificados com um sinal de (+). |
| urldecode() | Desfaz o efeito de urlencode(). |
| trim() | Remove os espaços do início e do fim do texto. |
| ltrim() | Remove os espaços do início do texto. |

| Função | Descrição |
|----------------------|--|
| chop() | Remove os espaços do fim do texto. |
| strstr() | Encontra a primeira ocorrência de uma string e retorna true caso encontre ou false, caso não encontre. |
| stristr() | Funciona como o strstr(), porém sem diferenciar maiúscula de minúsculas. |
| strlen() | Retorna o tamanho de uma string. |
| str_replace() | Substitui todas as ocorrências da string de procura com a string de substituição. |

As funções nos auxiliam muito na organização do código pois estes ficam mais limpos e ágeis. Então, aproveite e crie suas funções durante a criação dos scripts. Elas poderão ser reutilizadas e, ainda, facilitarão a manutenção no código.

Indicações



Para saber mais acerca das funções built-in, pesquise sobre:

- funções gettype(); is_numeric(), is_int(), is_integer(), is_real(), is_long(), is_float(), is_string(), is_array() e is_object(), isset(), empty().

Atividades



1. Como reconhecer uma variável no PHP?
2. Qual a expressão utilizada para imprimir um resultado na tela do computador?
3. Como utilizamos um array no PHP?
4. Sabendo da existência de um array bidimensional (conforme visto na figura do lado esquerdo), desenvolva uma tabela como a mostrada no lado direito:

```
$stabela[0][0] = "DVD";
$stabela[0][1] = "5";
$stabela[0][2] = "1.00";
$stabela[1][0] = "PenDrive 4GB";
$stabela[1][1] = "8";
$stabela[1][2] = "23.00";
$stabela[2][0] = "Mouse";
$stabela[2][1] = "10";
$stabela[2][2] = "15.00";
```

| Nome | Quant | Preço | Total |
|-----------------|-------|----------|-------------------------------------|
| DVD | 5 | R\$1,00 | R\$5,00 |
| PenDrive 4GB | 8 | R\$23,00 | R\$184,00 |
| Mouse | 10 | R\$15,00 | R\$150,00 |
| Total | | | R\$339,00 |
| AUXÍLIO: | | | number_format(\$valor, 2, ',', '.') |

5. Escreva um script contendo 5 variáveis com valores numéricos quaisquer e imprima o valor das 5 variáveis.

- Depois atribua à primeira variável a soma das outras quatro variáveis. Assim, imprima. Logo após, acrescente 1 ao valor da

segunda variável e decresça em 1 o valor da terceira variável. Assim, imprima. Por fim, acrescente à quarta variável o resto de sua divisão pela quinta variável.

6. Crie um script que contenha 5 variáveis numéricas. Teste para saber se cada uma delas é par e imprima o resultado na tela. Ex.: A variável \$a possui valor 5, sendo assim um número ímpar.
 7. Explique qual é a sintaxe das funções.
 8. Explique a diferença entre variável local e global.



Atividades



Anotações

3.14 INTERNACIONALIZAÇÃO DE FORMULÁRIOS HTML

Cada vez mais os sites necessitam do recurso de internacionalização dos formulários que utilizam, e quem os desenvolve com esse intuito já sai na frente, pois deixa seu site/sistema Web mais flexível, agilizando até mesmo as possíveis alterações.

A internacionalização possibilita que, com apenas um clique, o usuário mude o idioma do formulário.

O nosso objetivo aqui será criar dois arquivos .ini (um para cada nacionalidade) e oferecer a opção para o usuário escolher o idioma que melhor atendê-lo. Veja:

Crie um arquivo chamado en.ini. Esse arquivo será responsável por armazenar as informações relacionadas ao idioma inglês.

Figura 51 - Arquivo en.ini

```
1 titulo1 = Contact Form
2 titulo2 = Enter your details below
3 nome = Your name
4 telefone = Your Phone
5 email = Your e-mail
6 submit = Send Mensage
7 reset = Erase
```

Crie um arquivo chamado pt-br.ini. Esse arquivo será responsável por armazenar as informações relacionadas ao idioma português.

Figura 52 - Arquivo pt-br.ini

```
1 titulo1 = Formulário de Contato
2 titulo2 = Insira seus dados abaixo
3 nome = Seu nome
4 telefone = Seu Telefone
5 email = Seu E-mail
6 submit = Enviar Mensagem
7 reset = Apagar
```

Crie agora o seu arquivo de formulário, como o modelo abaixo:

Figura 53 - Arquivo contato.php

```

1 <?php
2 $idioma = $_GET["lan"]; // recebe o idioma escolhido
3 $dados = parse_ini_file($idioma.".ini"); // cria um array com os dados do arquivo .ini
4 ?>
5 <html>
6   <head>
7     <title><?php echo $dados['titulo1']; ?></title>
8   </head>
9   <body>
10    <a href="?lan=pt-br">BR&gt; - <a href="?lan=en">USA</a>
11    <form name="cadastro" action="formulario.php" method="post">
12      <table border="1" width="400">
13        <tr>
14          <td colspan="2" align="center"><?php echo $dados['titulo2']; ?></td>
15        </tr>
16        <tr>
17          <td><?php echo $dados['nome']; ?></td>
18          <td><input type="text" name="nome" value="" /></td>
19        </tr>
20        <tr>
21          <td><?php echo $dados['telefone']; ?></td>
22          <td><input type="text" name="telefone" value="" /></td>
23        </tr>
24        <tr>
25          <td><?php echo $dados['email']; ?></td>
26          <td><input type="text" name="email" value="" /></td>
27        </tr>
28        <tr>
29          <td colspan="2" align="center">
30            <input type="submit" name="enviar" value="<?php echo $dados['submit']; ?>" />
31            <input type="reset" name="apagar" value="<?php echo $dados['reset']; ?>" />
32          </td>
33        </tr>
34      </table>
35    </form>
36  </body>
37 </html>

```

É importante fazermos as seguintes observações:

- **Linha 2:** Recebemos o valor informado no link da linha 10.
- **Linha 3:** Cria-se um array com base no arquivo do idioma escolhida pelo usuário.
- **Linha 10:** Link para o usuário escolher entre o idioma português ou inglês.
- **Linha 07, 14, 17, 21, 25, 30 e 31:** São impressos os valores para os campos titulo1, titulo2, nome, telefone, email, submit e reset, respectivamente. Esses valores estão armazenados no array que os buscou no arquivo .ini.

3.15 PASSANDO INFORMAÇÕES ENTRE AS PÁGINAS

Um recurso muito importante nas páginas WEB é a passagem de informações entre as páginas. O envio de informações se dá por meio de duas maneiras: método GET e método POST. Apesar de não ser método, ainda existe outra maneira de transitar dados entre as páginas: via sessões (ver tópico 3.16).

Como já foi citado, o formulário serve para criar a interação com o usuário. As informações são enviadas pelo formulário através dos métodos GET ou POST.

Para auxiliar os programadores, o PHP fornece as variáveis de *arrays superglobais* (`$_GET` e `$_POST`) como recurso para a transferência dessas informações.

3.15.1 Método GET

Figura 54 - Arquivo formulario.html

```
1 <html>
2   <head>
3     <title>Meu formulário</title>
4   </head>
5   <body>
6     <form name="cadastro" action="procFormGet.php" method="get">
7       <table border="1" width="400">
8         <tr>
9           <td colspan="2" align="center">Formulário de Cadastro</td>
10          </tr>
11          <tr>
12            <td>Nome</td>
13            <td><input type="text" name="name" value="" /></td>
14          </tr>
15          <tr>
16            <td>Senha</td>
17            <td><input type="password" name="senha" value="" /></td>
18          </tr>
19          <tr>
20            <td colspan="2" align="center">
21              <input type="submit" name="enviar" value="Enviar">
22              <input type="reset" name="apagar" value="Apagar">
23            </td>
24          </tr>
25        </table>
26      </form>
27    </body>
28  </html>
```

Veja que, na linha 6, da figura 54, colocamos o “method” como “get” que tem por objetivo enviar os valores via barra de endereço do navegador. Veja abaixo o arquivo responsável pelo recebimento dos dados.

Observe que nas linhas 12 e 16, da figura 55, eu utilizei o *array superglobal* `$_GET`, sendo que dentro dos colchetes (`[]`) foi colocado o nome do campo do formulário.

Figura 55 - Arquivo procFormGet.php

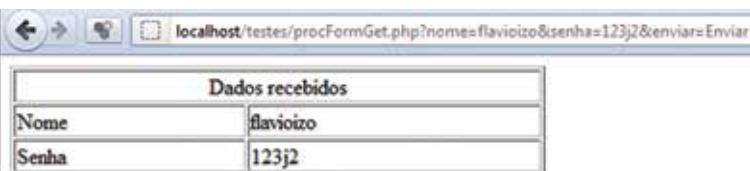
```

1  <html>
2  	<head>
3  		<title>Meu formulário</title>
4  	</head>
5  	<body>
6    	<table border="1" width="400">
7     	<tr>
8        	<td colspan="2" align="center">Dados recebidos</td>
9     	</tr>
10    	<tr>
11       	<td>Nome</td>
12       	<td><?php echo $_GET['nome']; ?></td>
13    	</tr>
14    	<tr>
15       	<td>Senha</td>
16       	<td><?php echo $_GET['senha']; ?></td>
17    	</tr>
18   	</table>
19  	</body>
20 </html>

```

Esse exemplo não seria muito adequado ser feito através do método get, haja vista que os valores digitados apareceriam na barra de endereços, e entre eles a senha do usuário. Veja:

Figura 56 - Arquivo procFormGet.php após interpretação



Perceba que a senha do usuário aparece, logo, pode ficar armazenada no histórico de páginas acessadas do navegador ou, ainda, alguém que estivesse por perto poderia visualizá-la.

A URL é formada colocando-se “?” após o nome da página e os campos (parâmetros) são separados pelo “&”.

Nessa situação, o mais adequado seria utilizar o método POST.

3.15.2 Método POST

O método POST funciona de maneira parecida, porém os dados do formulário são enviados de forma encapsulada e, consequentemente, não aparecem na barra de endereços.

Nós podemos utilizar o mesmo formulário da figura 54. No entanto, é necessário modificar o método para “post”. Também alteramos o *action* para uma página que receba os valores via post. (linha 6).

Figura 57 - Arquivo formularioPost.html

```
1  <html>
2   <head>
3     <title>Meu formulário</title>
4   </head>
5   <body>
6     <form name="cadastro" action="procFormPost.php" method="post">
7       <table border="1" width="400">
8         <tr>
9           <td colspan="2" align="center">Formulário de Cadastro</td>
10        </tr>
11        <tr>
12          <td>Nome</td>
13          <td><input type="text" name="nome" value="" /></td>
14        </tr>
15        <tr>
16          <td>Senha</td>
17          <td><input type="password" name="senha" value="" /></td>
18        </tr>
19        <tr>
20          <td colspan="2" align="center">
21            <input type="submit" name="enviar" value="Enviar">
22            <input type="reset" name="apagar" value="Apagar">
23          </td>
24        </tr>
25      </table>
26    </form>
27  </body>
28 </html>
```

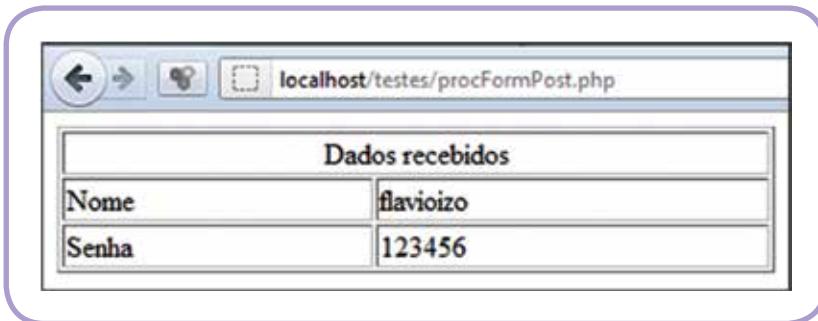
Agora criaremos a página que receberá os valores enviados via post. Observe que nas linhas 12 e 16, da figura 58, eu utilizei o *array superglobal* `$_POST`, sendo que dentro dos colchetes (`[]`) foi colocado o nome do campo do formulário.

Figura 58 - Arquivo procFormPost.php

```
<html>
  <head>
    <title>Meu formulário</title>
  </head>
  <body>
    <table border="1" width="400">
      <tr>
        <td colspan="2" align="center">Dados recebidos</td>
      </tr>
      <tr>
        <td>Nome</td>
        <td><?php echo $_POST['nome']; ?></td>
      </tr>
      <tr>
        <td>Senha</td>
        <td><?php echo $_POST['senha']; ?></td>
      </tr>
    </table>
  </body>
</html>
```

Dessa forma, quando a página procFormPost.php é chamada, os dados do formulário são passados normalmente, porém não aparecem na barra de endereços.

Figura 59 - Arquivo procFormGet.php após interpretação



Eis a pergunta: Qual dos dois métodos é o melhor?

Os dois são bons e úteis. Pense da seguinte forma: sempre que for buscar ou consultar alguma coisa que não vá alterar o estado do servidor, utilize *GET*; já se você for fazer alguma alteração com a requisição (gravação em BD, por exemplo), envio de arquivo ou os dados forem muitos, utilize *POST*.

3.16 SESSÕES

As sessões são fundamentais para quem quer ser desenvolvedor de páginas WEB, principalmente em relação a sistemas que necessitam de autenticação e *login*.

Observe essa situação:

- Ao acessar seu e-mail, você faz o *login*, correto? Após logar, você consegue navegar por entre as páginas sem precisar logar novamente. Isso acontece porque uma sessão é aberta e toda vez que uma página é acessada (seja uma mensagem, a área de spam, os e-mails apagados, etc.), é verificado se existe uma sessão aberta, se existir, o usuário pode navegar normalmente pelas páginas.

Você se familiarizou com essa situação? Aposto que sim, pois o uso de sessões é bem comum no nosso dia a dia.

As sessões são variáveis globais utilizadas com o auxílio do array `$_SESSION[]`. Uma vez criada uma sessão, esta ficará disponível durante todo o tempo de execução e poderá ser eliminada quando seu tempo de existência expirar ou quando a sessão for morta pelo usuário.

 Conceitos

Veja, abaixo, algumas funções para serem utilizadas com as sessões:

| Função | Descrição |
|-------------------------------|--|
| session_start() | Inicializa os dados de uma sessão. Deve ser colocado em todas as páginas que for utilizar sessão. OBS.: Deve ser colocado como a primeira linha da página. |
| session_destroy() | Destrói os dados de uma sessão, caso a sessão seja informada como parâmetro, ou destrói todas, caso nenhuma sessão seja informada como parâmetro. |
| session_cache_expire() | Retorna quando o cache atual expira. |
| session_unset() | Libera as variáveis de uma sessão, caso a sessão seja informada como parâmetro, ou libera todas, caso nenhuma sessão seja informada como parâmetro. |
| isset() | Verifica se uma variável foi registrada. |

Figura 60 - Utilizando sessões

```

1 <?php
2 // Inicializa a sessão
3 session_start();
4
5 // Registra ou incrementa o contador
6 if (!isset($_SESSION['contador'])) {
7     $_SESSION['contador'] = 1;
8     $_SESSION['inicio'] = time();
9 } else {
10     $_SESSION['contador'] = $_SESSION['contador'] + 1;
11 }
12 ?>
13
14 <html>
15 <head>
16     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
17     <title>Sessões</title>
18 </head>
19 <body>
20     <p>Contador = <?php echo $_SESSION["contador"]; ?></p>
21     <p>A sessão está aberta a <?php echo time() - $_SESSION["inicio"]; ?> segundos.</p>
22 </body>
23 </html>

```

Façamos as explicações:

Na linha 3, informei que irei utilizar sessão nesta página. Repare que foi o primeiro comando da programação.

Na linha 6 utilizei a função `isset()` para testar se a variável `$_SESSION["contador"]` já tinha sido “setada” (declarada). Caso a sessão não tenha sido declarada ainda, uma sessão “contador” é iniciada e uma sessão `início` é criada também para sabermos em que momento a sessão foi criada.

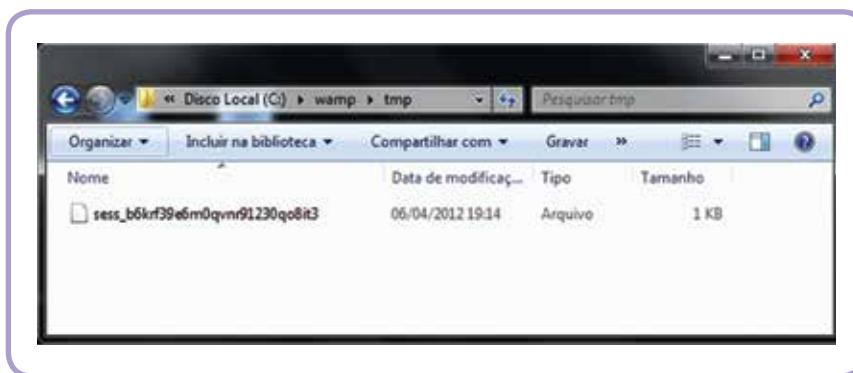
Na linha 10, caso a sessão “contador” já tenha sido inicializada, a sessão “contador” será incrementada em 1 unidade, ou seja, toda vez que atualizar a página, será incrementado 1 na variável \$_SESSION[“contador”].

[Linhas 20 e 21](#), faço a impressão dos valores armazenados nas variáveis de sessão.

Toda vez que uma sessão é criada, esta fica armazenada em forma de arquivo na pasta de configuração de sessões. Para saber onde suas sessões ficam armazenadas, você deve procurar pela variável session.save_path dentro do php.ini, conforme vimos no item 3.2 Configurações do PHP.

No meu caso, ao executar a página php do exemplo anterior, foi gerado um arquivo de sessão, conforme mostra figura abaixo:

Figura 61 - Pasta de armazenamento de sessões



Mais a frente, veremos um exemplo completo com sistema de login e autenticação.

3.17 UPLOAD DE ARQUIVOS

O upload de arquivo é um recurso muito utilizado em sites e/ou sistemas WEB. Pense, por exemplo, na sua conta de e-mail. Você já deve ter utilizado o envio de anexo, correto? Então, ao enviar um anexo é feito um upload do arquivo para o servidor de e-mail e enviado ao destinatário.

Apesar de ser simples, o upload de arquivos possui algumas regras que normalmente ocasionam grandes dores de cabeça quando são esquecidas pelo programador. Vejamos:

| Regra | Erro quando não seguida |
|--|---|
| Enctype do formulário deve estar como “multipart/form-data”. | O arquivo não será convertido para binário e, consequentemente, não será enviado para a página definida no <i>action</i> do formulário. |
| Método do formulário deve ser “post”. | Somente o método post consegue enviar dados, então o envio de dados não será feito, somente envio de texto. |
| Tipo do campo do formulário de envio dever ser “file” | Não vai enviar o arquivo, somente o valor contido no campo. |
| O diretório que vai receber o arquivo deve ter permissão de gravação. | O arquivo não será copiado para a pasta, haja vista que não terá permissão. |

Atenção

Apesar de não ser regra, uma dica é que você não permita que o usuário faça envio de arquivos com extensão .php para o servidor, pois um usuário mal intencionado poderá enviar um código malicioso para o servidor de hospedagem e, ao executá-lo, poderá ter acesso a dados pessoais ou mesmo danificar seu site/sistema.

Para efetuar um upload, você poderá utilizar duas funções:

- **copy()**: não permitida na maioria dos servidores de hospedagem, pois simplesmente copia um arquivo, sem efetuar verificações na estrutura.
- **move_uploaded_file()**: mais utilizado, haja vista que esta função faz uma verificação para ter certeza de que o arquivo designado é um arquivo de upload válido (que tenha sido enviado pelo método de envio POST HTTP). Se o arquivo for válido, ele será movido.

A seguir, foi colocado um exemplo bem simples de upload de arquivo. Esse exemplo pode ser aperfeiçoadinho por você.

Ao aprender a fazer uploads de arquivos, comece a pensar em situações nas quais você possa utilizar o upload.

Programação Web

Figura 62 - Página formUpload.html

```
1[<] <html>
2[<]   <head>
3[<]     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
4[<]     <script type="text/javascript">
5[<]       <!--
6[<]       function teste(){
7[<]         if (document.upload.arquivo.value=="") {
8[<]           alert("Arquivo para upload não informado!");
9[<]           document.upload.arquivo.focus();
10[<]          return false;
11[<]        }
12[<]      }-->
13[<]     </script>
14[<]   <title>Upload simples (copy e move_uploaded_file)</title>
15[<] </head>
16[<] <body>
17[<]   <h2>Upload Simples</h2><br>
18[<]
19[<]   <form action="upload.php" name="formUpload" method="post" enctype="multipart/form-data"
20[<]     onsubmit="return teste();">
21[<]       <p><input type="file" name="arquivo" size="60"></p>
22[<]
23[<]
24[<]       <p>
25[<]         Fun&ccedil;&atilde;o a ser usada para fazer o upload do arquivo:<br />
26[<]         <input type="radio" name="radio" value="copy" checked>copy();<br />
27[<]         <input type="radio" name="radio" value="move_uploaded_file">move_uploaded_file();
28[<]       </p>
29[<]
30[<]       <p>
31[<]         <input type="submit" name="enviar" value="Upload!"><br />
32[<]         Por razões de segurança, no nosso exemplo não é permitido o envio
33[<]         de arquivos do tipo .php<br />
34[<]       </p>
35[<]       <br />
36[<]       <p>
37[<]         <strong>A função move_uploaded_file():</strong>
38[<]         Esta função verifica para ter certeza de que o arquivo designado é
39[<]         um arquivo de upload válido (que tenha sido enviado pelo mecanismo
40[<]         PHP de envio por POST HTTP). Se o arquivo for válido, ele será movido.
41[<]         <br />
42[<]         <strong>A função copy():</strong>
43[<]         O copy simplesmente o copia, não faz verificação alguma.
44[<]         Não é aceita em alguns servidores WEB.
45[<]       </p>
46[<]     </form>
47[<]   </body>
48[<] </html>
49[<] </body>
50[<] </html>
```

Figura 63 - Página upload.php

```

1 <?php
2 /* Verifica o tamanho do arquivo só pra saber se é bytes, KB ou MB */
3 if (!isset($_FILES['arquivo'])) {
4     if ($_FILES['arquivo']['size'] > 1024 * 1024) {
5         $tamanho = round($_FILES['arquivo']['size'] / 1024 / 1024, 2);
6         $med = "MB";
7     } else if ($_FILES['arquivo']['size'] > 1024) {
8         $tamanho = round($_FILES['arquivo']['size'] / 1024, 2);
9         $med = "KB";
10    } else {
11        $tamanho = $_FILES['arquivo']['size'];
12        $med = "Bytes";
13    }
14
15 /* Informações do tamanho máximo do arquivo em bytes: */
16 echo "Tamanho: ". $tamanho."$med";
17 if($_FILES['arquivo']['size'] > 5242880) { //Limite: 5MB
18     echo "<script> alert('Tamanho: $tamanho $med! Seu arquivo não poderá ser
19         maior que 5MB!'); window.history.go(-1); </script>\n";
20     exit;
21 }
22
23 /* Informações do diretório destino do upload */
24 if (!is_file($_FILES['arquivo']['tmp_name'])) {
25     $arquivo = $_FILES['arquivo']['tmp_name'];
26     $caminho = "./arquivos/";
27     $scaminho = $caminho.$_FILES['arquivo']['name'];
28
29     /* Informações do tipo de arquivo suportado */
30     if (!preg_match("/\.(php|js|css|png|jpg|jpeg|gif|pdf|zip|tar|tar.gz|rar)/", $_FILES['arquivo']['name'])) {
31         if ($_POST['radio'] == "copy") {
32             copy($arquivo,$scaminho) or die("<p>Erro durante a manipulação do arquivo '$arquivo'</p>
33             .<p><a href='".$SERVER["PHP_SELF"]."'>Voltar</a></p>");
34         } else if ($_POST['radio'] == "move_uploaded_file") {
35             move_uploaded_file($arquivo,$scaminho) or die("<p>Erro durante a manipulação do
36                 arquivo '$arquivo'</p> .<p><a href='".$SERVER["PHP_SELF"]."'>Voltar</a></p>");
37         }
38         echo "<h3><center>Arquivo enviado com sucesso usando " . $_POST['radio'] . "</center></h3>";
39     } else {
40         echo "<h3><center>Arquivo n&atilde;o enviado!</center></h3>\n";
41         echo "<h4><font color='FF0000'><center>Caminho ou nome de arquivo inv&acirc;lido!</center></font></h4>";
42     }
43 }
44 echo("Local Temp: $arquivo<br />Destino: $caminho [<a href='".$scaminho."'>Download</a>]");
45 }
?>

```

3.18 PROBLEMAS COMUNS COM PHP

Abaixo, foram selecionados alguns erros que normalmente acontecem ao se desenvolver códigos em PHP.

Os códigos PHP aparecem como se fossem HTML ou o navegador pede para salvar o arquivo

- O mecanismo PHP não está sendo invocado corretamente. Tente verificar se você está chamando o script como informado anteriormente, utilizando o `http://localhost/sua_página`
- Verifique, também, se o servidor web foi iniciado corretamente.

A página não pode ser exibida

- Reinicie o servidor e verifique se o servidor web foi iniciado corretamente.

A página aparece totalmente em branco

- Esse pode ser um típico erro de HTML. Por isso, verifique se você deixou alguma tag aberta, por exemplo <form>.

Parte do código PHP aparece na tela

- Você, provavelmente, esqueceu de fechar a tag do PHP “?>”.

Erro de análise sintática (*Parse error*)

- Não foi colocado “;” no final de cada linha de instrução.
- Ausência do sinal de cifrão no início das variáveis.
- Não fechamento do código PHP utilizando o “?>”.
- Má utilização de aspas.

Erro 403

- Esse erro normalmente está relacionado à falta de permissão para utilização do arquivo.

Erro de *timeout*: (*Maximum execution time of 300 seconds exceeded*)

- Seu script está demorando a ser executado e ultrapassou o limite definido no php.ini. Provavelmente, deve estar em loop infinito ou faltando alguma “}”.

1. Explique, em linhas gerais, o que é e como é o funcionamento da internacionalização de formulários.

2. Quais os dois métodos do formulário que nos permitem enviar os dados para outra página? Explique as diferenças entre eles. Qual dos dois é o melhor método?

3. O que são sessões?

4. Explique para que servem as funções session_start() e session_destroy().

5. Quais são as regras necessárias para elaborar um script que faça upload de arquivos?

6. É possível e interessante disponibilizar o upload de qualquer tipo de arquivo?

7. Quais as funções que permitem copiar um arquivo para um diretório do servidor? Diferencie-as.



Atividades

Anotações



UTILIZANDO O BANCO DE DADOS

Olá!

Neste quarto capítulo falaremos acerca da comunicação do PHP com o Banco de Dados.

O objetivo é estudar como estabelecer uma conexão com o banco, assim como verificar a inserção, deleção e atualização de informações através de formulários.

Também veremos como podemos resgatar as informações armazenadas no banco de dados e exibi-las na tela.

Os conhecimentos adquiridos neste capítulo serão úteis para a aprendizagem dos demais conteúdos trabalhados nesta disciplina.

Bons estudos!

Angue risus at
e velit at tellus.
massa portitor
sectetur magna.

Fala Professor

Por melhor que seja a linguagem de programação, sem a comunicação com um banco de dados ela não representará muita coisa. O banco de dados é essencial para que os dados possam ser armazenados e serem reutilizados em uma consulta futura.

4.1 FERRAMENTAS PARA ADMINISTRAR O BANCO DE DADOS

Existem diversas ferramentas disponíveis para a administração de banco de dados. Eu irei citar duas:

| | | |
|------------|------------|---|
| MySQL | PHPMyAdmin | http://www.phpmyadmin.net/home_page/downloads.php |
| PostgreSQL | PgAdmin | http://www.pgadmin.org/download/ |

É importante citar que, ao instalar um pacote utilitário (Wamp, PHPtriad etc.), automaticamente já será instalado o PHPMyAdmin. O mesmo acontece quando instalamos o PostgreSQL, sendo que este instala o PgAdmin.

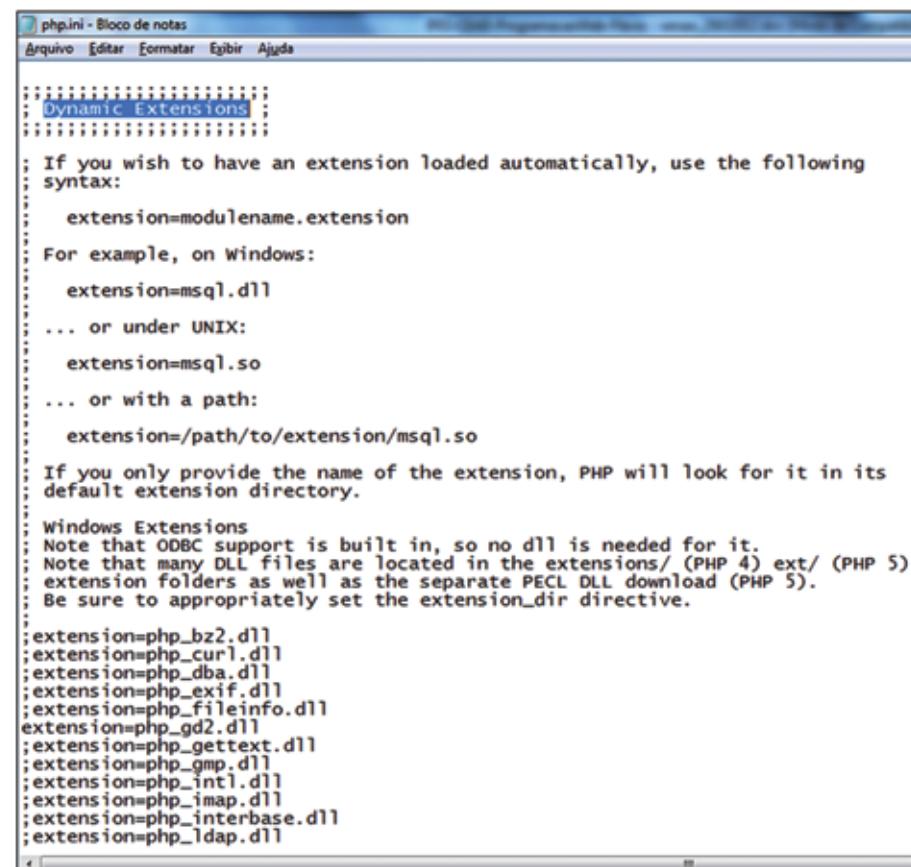
Para quem vai utilizar o phpmyadmin, deve acessar o link <http://127.0.0.1/phpmyadmin>

Para quem vai utilizar o pgAdmin, deve acessar o programa que normalmente fica instalado em “C:\Program Files\PostgreSQL\9.0\bin\pgAdmin3.exe”.

4.2 HABILITANDO UMA CONEXÃO

O PHP conecta-se a vários bancos de dados. Para habilitá-lo, você deve acessar o arquivo php.ini e procurar pela referência “*Dynamic Extensions*”. Essas extensões possibilitam que recursos adicionais sejam acessados pelo PHP, entre eles o acesso a alguns bancos de dados.

Figura 64 - Visualizando o php.ini



```
Dynamic Extensions
-----
; If you wish to have an extension loaded automatically, use the following
; syntax:
; extension=modulename.extension
;
; For example, on Windows:
; extension=msql.dll
;
; ... or under UNIX:
; extension=msql.so
;
; ... or with a path:
; extension=/path/to/extension/msql.so
;
; If you only provide the name of the extension, PHP will look for it in its
; default extension directory.
;
; Windows Extensions
; Note that ODBC support is built in, so no dll is needed for it.
; Note that many DLL files are located in the extensions/ (PHP 4) ext/ (PHP 5)
; extension folders as well as the separate PECL DLL download (PHP 5).
; Be sure to appropriately set the extension_dir directive.
;
extension=php_bz2.dll
extension=php_curl.dll
extension=php_dba.dll
extension=php_exif.dll
extension=php_fileinfo.dll
extension=php_gd2.dll
extension=php_gettext.dll
extension=php_gmp.dll
extension=php_intl.dll
extension=php_imap.dll
extension=php_interbase.dll
extension=php_ldap.dll
```

Como visto na parte de baixo do php.ini, algumas das extensões aparecem com um “;” (ponto e vírgula) antes do nome da extensão. Isso indica que a extensão está desabilitada. Segundo esse raciocínio, a única extensão mostrada na figura 64 que está habilitada é a “php_gd2.dll”.

Analisando mais abaixo do php.ini, veremos que existem extensões relativas aos banco de dados. Veja na figura:

Figura 65 - Visualizando o php.ini

```
;extension=php_interbase.dll
;extension=php_ldap.dll
extension=php_mbstring.dll
;extension=php_ming.dll
;extension=php_mssql.dll
extension=php_mysql.dll
extension=php=mysqli.dll
;extension=php_ocis.dll      ; Use with Oracle 10gR2 Instant Client
;extension=php_ocis_11g.dll  ; Use with Oracle 11g Instant Client
;extension=php_openssl.dll
;extension=php_pdo_firebird.dll
;extension=php_pdo_mssql.dll
extension=php_pdo_mysql.dll
;extension=php_pdo_oci.dll
;extension=php_pdo_odbc.dll
;extension=php_pdo_pgsql.dll
extension=php_pdo_sqlite.dll
extension=php_pgsql.dll
;extension=php_phar.dll
;extension=php_ps.dll
```

Assim, percebemos que tanto o PostgreSQL (php_pgsql.dll) quanto o Mysql (php_mysql.dll) estão habilitados. Caso você queira habilitar outra extensão, você deve retirar o “;” salvar o arquivo php.ini e reiniciar seu servidor web.

Para que o PostgreSQL funcione, além de habilitar a extensão “php_pgsql.dll”, você também necessita copiar a dll “libpq.dll” da pasta de instalação do PHP para a pasta bin de instalação do servidor web.

Ex.: Quem utiliza o Wamp Server deve:

- copiar de “C:\wamp\bin\php\php5.3.0”
- para “C:\wamp\bin\apache\Apache2.2.11\bin”

Atenção

4.3 COMUNICAÇÃO DA LINGUAGEM COM O BANCO DE DADOS

Vamos utilizar como exemplo o banco de dados MySql, porém a ideologia é parecida para qualquer banco de dados, mudando somente o nome das funções.

4.3.1 Conectando ao banco de dados MySql

A função utilizada para iniciar uma conexão com o MySql é:

```
mysql_connect(servidor,usuário,senha)
```

Onde:

SERVIDOR é o servidor no qual o banco está instalado.
Normalmente é localhost ou 127.0.0.1

USUÁRIO é o nome do usuário utilizado para conectar-se ao BD.

SENHA é a senha utilizada para conectar-se ao BD.

Ao chamar a função, o retorno será um identificador da conexão ou false, em caso de erro. Deve-se armazenar a conexão em uma variável, tendo em vista que posteriormente poderemos utilizar o identificador da conexão. Veja abaixo um exemplo de conexão:

```
$con = mysql_connect("127.0.0.1","teste","senha123");
```

4.3.2 Selecionando um banco de dados

Um servidor de banco de dados pode ter vários bancos instalados. Sendo assim, além de estabelecer uma conexão, é necessário selecionar o banco de dados de onde iremos gravar ou selecionar os dados.

Para selecionar um BD, será necessário utilizar a função

```
mysql_select_db(BancoDeDados, CONEXÃO)
```

Onde:

BancoDeDados é o nome do banco que será selecionado,

CONEXÃO é o identificador da conexão aberta (opcional),

A conexão é opcional pois, se o usuário não informá-la, a função irá utilizar a última conexão aberta. Vejamos um exemplo:

```
$bd = mysql_select_db("ifes",$con);
```

4.3.3 Executando uma query

Para executar uma query, é necessário utilizar a função

```
mysql_query(query, CONEXÃO)
```

Onde:

QUERY é o sql a ser executado (pode ser um insert, update, delete, select etc.).

CONEXÃO é o identificador da conexão aberta (opcional).

A conexão é opcional pois, se o usuário não informá-la, a função irá utilizar a última conexão aberta. Vejamos alguns exemplos:

```
$sql = mysql_query("select * from pw.usuarios",$con);

$sql = mysql_query("UPDATE pw.usuarios SET nome = 'Flávio Izo'
WHERE usuarios.cod_user =1",$con);

$sql = mysql_query("INSERT INTO pw.usuarios ('cod_user', 'nome',
'login', 'senha', 'status') VALUES (NULL, 'Flávio Izo', 'fizo', '12345',
'1');",$con);
```

4.3.4 Organizando uma consulta

Quando executamos uma consulta ao BD, todas as informações ficam armazenadas em uma variável. Sendo assim, essa variável deverá ser um *array*. No exemplo anterior, armazenamos o resultado da consulta na variável \$sql. Vejamos algumas funções que podem nos auxiliar na organização de nossas consultas:

mysql_num_rows(\$resultadoDaQuery):

Esta função retorna o número de linhas encontradas na execução da query.

Onde:

\$resultadoDaQuery é a variável que possui o resultado da consulta.

mysql_affected_rows(CONEXÃO):

Esta função retorna o número de linhas afetadas por uma alteração (update, delete ou insert).

Onde:

CONEXÃO é o identificador da conexão aberta (opcional).

A conexão é opcional pois, se o usuário não informá-la, a função irá utilizar como parâmetro a última conexão aberta.

mysql_fetch_row(\$resultadoDaQuery):

Esta função retorna cada coluna de cada linha do resultado da query de forma numérica.

Onde:

\$resultadoDaQuery é a variável que possui o resultado da consulta.

Veja um exemplo:

Figura 66 - Exemplo de consulta

```
1 <?php
2 $con = mysql_connect("localhost", "admin", "senha123");
3
4 $bd = mysql_select_db("pw", $con);
5
6 $busca = mysql_query("select * from pw.usuarios") or die("Erro:" .mysql_error());
7
8 $num_busca = mysql_num_rows($busca) or die("Erro:" .mysql_error());
9
10 for($i=0;$i<$num_busca;$i++){
11     $resultado = mysql_fetch_row($busca);
12     echo $resultado[0];
13     echo $resultado[1];
14     echo $resultado[2];
15     echo $resultado[3] . "<br />";
16 }
17
18 ?>
```

Façamos um breve resumo do que está acontecendo no exemplo acima:

Linha 02: conectamos com o servidor do MySql;

Linha 04: selecionamos o banco de dados da conexão \$con;

Linha 06: executamos a query de consulta ao banco;

Linha 08: recebemos o número de linhas retornadas na query;

Linhas 10 a 16: imprimimos todas as linhas retornadas na consulta.;

Linha 11: aqui armazenamos na variável \$resultado o array do tipo numérico contendo os dados da consulta. Perceba que. a cada chamada da função mysql_fetch_row(). armazenamos uma linha de resultado.

Linhas 12 a 15: são impressas as colunas relacionadas à linha de dados atual. A sequência numérica do array é de acordo com a sequência das colunas na tabela do banco de dados.

mysql_fetch_assoc(\$resultadoDaQuery):

Esta função retorna cada coluna de cada linha do resultado da query de forma associativa.

Onde:

\$resultadoDaQuery é a variável que possui o resultado da consulta.

Veja um exemplo:

Figura 67 - Exemplo de consulta

```

1 <?php
2 $con = mysql_connect("localhost","admin","senha123");
3
4 $bd = mysql_select_db("pw",$con);
5
6 $busca = mysql_query("select * from pw.usuarios") or die("Erro:".mysql_error());
7
8 $num_busca = mysql_num_rows($busca) or die("Erro:".mysql_error());
9
10 for($i=0;$i<$num_busca;$i++){
11     $resultado = mysql_fetch_assoc($busca);
12     echo $resultado['cod_user'];
13     echo $resultado['nome'];
14     echo $resultado['login'];
15     echo $resultado['senha']."<br />";
16 }
17 ?>

```

Façamos um breve resumo do que está acontecendo no exemplo acima, porém iremos considerar somente das linhas 11 a 15, pois o restante é igual ao exemplo anterior:

Linha 11: Aqui armazenamos na variável \$resultado o array do tipo associativo contendo os dados da consulta. Perceba que a cada chamada da função mysql_fetch_assoc() armazenamos uma linha de resultado.

Linhas 12 a 15: São impressas as colunas relacionadas à linha de dados atual. A sequência exibida é de acordo com o nome das colunas na tabela do banco de dados.

mysql_fetch_array(\$resultadoDaQuery,TIPO):

Esta função retorna cada coluna de cada linha do resultado da query de forma numérica, de forma associativa, ou as duas ao mesmo tempo.

Onde:

\$resultadoDaQuery é a variável que possui o resultado da consulta.

TIPO pode ser:

MYSQL_ASSOC é idêntico à função mysql_fetch_assoc, ou seja, retorna como índice associativo. Ex.: \$resultado['cod_user'];

MYSQL_NUM é idêntico à função mysql_fetch_row, ou seja, retorna como índice numérico. Ex.: \$resultado[0];

MYSQL_BOTH retorna como índice associativo e numérico.

Veja um exemplo:

Figura 68 - Exemplo de consulta

```
1 <?php
2   $con = mysql_connect("localhost", "admin", "senha123");
3
4   $bd = mysql_select_db("pw", $con);
5
6   $busca = mysql_query("select * from pw.usuarios") or die("Erro:" .mysql_error());
7
8   $num_busca = mysql_num_rows($busca) or die("Erro:" .mysql_error());
9
10  for($i=0;$i<$num_busca;$i++) {
11    $resultado = mysql_fetch_array($busca, MYSQL_BOTH);
12    echo $resultado[0];
13    echo $resultado['nome'];
14    echo $resultado['login'];
15    echo $resultado[3] ."  

```

Façamos um breve resumo do que está acontecendo no exemplo acima, porém iremos considerar somente das linhas 11 a 15, pois o restante é igual ao exemplo anterior:

Linha 11: Aqui armazenamos na variável \$resultado o array do tipo MYSQL_BOTH que pode ser chamado tanto de forma associativa ou numérica. Esse array contém os dados da consulta. Perceba que a cada chamada da função mysql_fetch_array() armazenamos uma linha de resultado.

Linhos 12 a 15: São impressas as colunas relacionadas à linha de dados atual. A sequência exibida pode ser de acordo com o nome das colunas na tabela do banco de dados ou de acordo com a sequência numérica.

mysql_fetch_object(\$resultadoDaQuery):

Esta função retorna como se fosse um **objeto**.

Onde:

\$resultadoDaQuery é a variável que possui o resultado da consulta.

Veja um exemplo:

Figura 69 - Exemplo de consulta

```

1 <?php
2 $con = mysql_connect("localhost", "admin", "senha123");
3
4 $bd = mysql_select_db("pw", $con);
5
6 $busca = mysql_query("select * from pw.usuarios") or die("Erro:" .mysql_error());
7
8 $num_busca = mysql_num_rows($busca) or die("Erro:" .mysql_error());
9
10 for($i=0;$i<$num_busca;$i++) {
11     $resultado = mysql_fetch_object($busca);
12     echo $resultado->cod_user;
13     echo $resultado->nome;
14     echo $resultado->login;
15     echo $resultado->senha . "<br />";
16 }
17
18 ?>

```

Façamos um breve resumo do que está acontecendo no exemplo acima, porém iremos considerar somente das linhas 11 a 15, pois o restante é igual ao exemplo anterior:

Linha 11: Aqui armazenamos na variável \$resultado o array do tipo object contendo os dados da consulta. Perceba que a cada chamada da função mysql_fetch_object(), armazenamos uma linha de resultado.

Linhas 12 a 15: São impressas as colunas relacionadas à linha de dados atual. A sequência exibida é de acordo com o nome das colunas na tabela do banco de dados.

mysql_result(\$resultadoDaQuery,Posição,Campo):

Esta função retorna o conteúdo de uma linha do resultado da query.

Onde:

\$resultadoDaQuery é a variável que possui o resultado da consulta.

Posição é a posição da linha de registro que se deseja retornar.

Campo é a coluna (campo) que se deseja retornar. Também é possível utilizar o índice ao invés do nome do campo, dessa forma o retorno fica mais rápido.

Veja um exemplo:

Figura 70 - Exemplo de consulta

```
1 <?php
2 $con = mysql_connect("localhost","admin","senha123");
3
4 $bd = mysql_select_db("pw",$con);
5
6 $busca = mysql_query("select * from pw.usuarios") or die("Erro:" .mysql_error());
7
8 $num_busca = mysql_num_rows($busca) or die("Erro:" .mysql_error());
9
10 echo mysql_result($busca,3,"cod_user");
11 echo mysql_result($busca,3,1);
12 echo mysql_result($busca,3,"login");
13 echo mysql_result($busca,3,"senha") . "<br />";
14
15 ?>
```

Façamos um breve resumo do que está acontecendo no exemplo acima, porém iremos considerar somente das linhas 10 a 13, pois o restante é igual aos exemplos anteriores:

Linhos 10 a 13: São impressas as colunas relacionadas à linha de dados atual. Veja que estamos imprimindo somente a linha de registro 4, haja vista que é um array iniciando-se do 0. Na linha 11, estamos utilizando o índice 1 ao invés do nome da coluna “nome”.

4.3.5 Fechando uma conexão

Para fechar uma conexão, é necessário utilizar a função

`mysql_close(CONEXÃO)`

Onde:

CONEXÃO é o identificador da conexão aberta (opcional).

Normalmente, a função `mysql_close()` não é utilizada, pois as conexões não são persistentes e se fecham ao final do script. Será importante utilizá-la quando fizermos uma conexão persistente (para isso utilize a função `mysql_pconnect()`).

Vejamos um exemplo:

```
$fechar = mysql_close($con);
```

4.3.6 Verificando erros

Você deve ter reparado que nós utilizamos uma função chamada `mysql_error()`. Essa função retorna uma mensagem de acordo com o erro encontrado, e pode ser utilizada combinada com a função `die()`.

Nos exemplos anteriores, sempre que chamávamos uma função de BD nós utilizamos o “`or die("Erro: " .mysql_error())`”. Ou seja, se a função retornasse algum erro, uma mensagem de erro seria exibida.

4.3.7 Outras funções do MySql

Abaixo, coloquei outras funções relacionadas ao banco de dados.

| Função | Descrição |
|------------------------------------|--|
| <code>mysql_create_db()</code> | Cria um novo banco de dados. |
| <code>mysql_drop_db()</code> | Apaga um banco de dados. |
| <code>mysql_errno()</code> | Retorna o número do erro. |
| <code>mysql_fetch_lengths()</code> | Retorna o comprimento de cada campo de um conjunto de resultados. |
| <code>mysql_fetch_field()</code> | Retorna informações sobre o campo. |
| <code>mysql_field_name()</code> | Retorna o nome do campo. |
| <code>mysql_field_table()</code> | Retorna o nome da tabela do campo informado. |
| <code>mysql_field_type()</code> | Retorna o tipo do campo informado. |
| <code>mysql_field_len()</code> | Retorna o comprimento do campo identificado. |
| <code>mysql_free_result()</code> | Libera a memória do resultado. |
| <code>mysql_insert_id()</code> | Retorna o ID da última inserção ou retorna false caso a última consulta não tenha sido uma inserção. |

Para saber mais sobre as funções de conexão, acesse os links abaixo:

- http://php.net/manual/pt_BR/function.mysql-connect.php
- http://php.net/manual/pt_BR/function.mysql-select-db.php
- http://php.net/manual/pt_BR/function.mysql-query.php
- http://php.net/manual/pt_BR/function.mysql-close.php
- http://php.net/manual/pt_BR/function.mysql-num-rows.php
- http://php.net/manual/pt_BR/function.mysql-affected-rows.php
- http://php.net/manual/pt_BR/function.mysql-fetch-row.php
- http://php.net/manual/pt_BR/function.mysql-fetch-assoc.php
- http://php.net/manual/pt_BR/function.mysql-fetch-array.php
- http://php.net/manual/pt_BR/function.mysql-fetch-object.php



Indicações

No capítulo 5, faremos uma implementação de estudo de caso no qual abordaremos um script que engloba sessão e conexão com banco de dados.

Atividades



1. Qual a importância do banco de dados para as linguagens de programação?
 2. Quais são os passos para habilitar uma conexão com o Banco de dados no PHP?
 3. É possível ter dois bancos de dados habilitados ao mesmo tempo?
 4. Considerando o mysql como o banco de dados a ser utilizado, quais as funções que nos permitem conectar e executar uma query?
 5. Considerando o mysql como o banco de dados a ser utilizado, como podemos organizar uma consulta?
 6. É necessário sempre fechar uma conexão com o banco de dados? Explique.
 7. Qual é o processo para verificar um erro vindo da execução de um comando do banco de dados? Dê um exemplo.

Anotações



IMPLEMENTAÇÃO DE ESTUDO DE CASO

Olá!

Neste quinto capítulo faremos um exemplo de um sistema simples, com as seguintes funcionalidades:

- autenticação de segurança com sessão;
- cadastro, alteração e exclusão de usuários.

Sendo assim, a partir deste estudo de caso, você poderá aplicar os conhecimentos aqui adquiridos em outros projetos que serão desenvolvidos no futuro.

Esse capítulo irá utilizar todos os conhecimentos adquiridos nos outros capítulos.

Bons estudos!

Inque ricus at
e velit at tellus.
nassa porttitor
issectetur magna.

Fala Professor

Para que nosso exemplo seja bem executado por você, seguem algumas informações das minhas configurações:

- Banco de dados utilizado: MySQL
- Usuário: root
- Senha:
- Nome do banco de dados: "ifes"
- Tabela utilizada:

| usuários | |
|----------|--------------------|
| ! | idUser: INTEGER |
| ◆ | nome: VARCHAR(50) |
| ◆ | login: VARCHAR(20) |
| ◆ | senha: VARCHAR(40) |
| ◆ | situacao: CHAR(1) |

Arquivo conexao.php

Figura 71 - Arquivo conexao.php

```
1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6
7 $local_serve = "localhost"; // local do servidor
8 $usuario_serve = "root"; // nome do usuário
9 $senha_serve = ""; // senha
10 $banco_de_dados = "ifes"; // nome do banco de dados
11
12 $conn = mysql_connect($local_serve,$usuario_serve,$senha_serve)
13 or die ("O servidor não responde!");
14
15 // conecta-se ao banco de dados
16 $db = mysql_select_db($banco_de_dados,$conn)
17 or die ("Erro ao selecionar o BD");
18 ?>
```

Arquivo login.php

Figura 72 - Arquivo login.php

```
1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6 ?
7 <html>
8   <head>
9     <title>Tela de Autenticação</title>
10    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
11  </head>
12  <body>
13    <form name="form1" method="POST" action="procLogin.php" enctype="x-www-form-urlencoded">
14
15      <table width="300" border="1" align="center" cellpadding="5" cellspacing="0">
16        <tr>
17          <td colspan="2" align="center" bgcolor="#E6E6FA">Autenticação</td>
18        </tr>
19        <tr>
20          <td>Login</td>
21          <td><input name="login" type="text" size="30" /></td>
22        </tr>
23        <tr>
24          <td>Senha</td>
25          <td><input name="senha" type="password" size="30" /></td>
26        </tr>
27        <tr>
28          <td colspan="2" align="center">
29            <input name="Submit" value="Logar" type="submit" />
30          </td>
31        </tr>
32      </table>
33    </form>
34  </body>
35 </html>
```

Veja que o campo senha é do tipo “password” (linha 25).

Arquivo procLogin.php

Figura 73 - Arquivo procLogin.php

```

1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6 session_start(); // informa que trabalharei com sessão nesta página
7 require("conexao.php"); //chama o arquivo de conexão com o BD
8
9 $login = $_POST['login'];
10 $senha = md5($_POST['senha']); // md5 é uma função que criptografa a senha
11
12 // Evita SQL injection, pois coloca / antes das aspas
13 $login = addslashes($login);
14 $senha = addslashes($senha);
15
16 // verificamos se o usuário e senha existem e se o usuário está ativo
17 $sql = "SELECT * FROM usuarios WHERE login='$login' AND senha='$senha' AND situacao='1'";
18 $resultado = mysql_query($sql) or die("ERRO NO COMANDO SQL");
19 if (mysql_num_rows($resultado)!=0) {
20     $_SESSION['sLogin']=$login; // criamos a sessão para o usuário
21
22     //redirecionamos para a página index
23     header("Location: index.php");
24 }else {
25     echo "<center>Erro: Usuário e/ou Senha inválidos</center><br />";
26     echo "<center><a href='login.php'>voltar</a></center>";
27 }
28 ?>

```

Observe que:

- Na linha 07, fizemos uma inclusão do arquivo “conexao.php” para que a conexão fosse estabelecida.
- Na linha 10, utilizamos a criptografia md5(), pois a senha está criptografada no BD.
- Nas linhas 13 e 14, utilizamos a função addslashes() para evitar situações de invasão por SQL Injection.
- Na linha 20, criamos uma sessão para o usuário, haja vista que o login e senha estavam corretos. Isso evita que a cada transição de página o usuário necessite digitar o login e senha novamente. Também vai evitar que um usuário acesse uma página de forma indevida.

Arquivo validaSessao.php

Figura 74 - Arquivo validaSessao.php

```
1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6
7 session_start(); // informa que trabalharei com sessão nesta página
8
9 if(!isset($_SESSION['sLogin'])) {
10     //redirecionamos para a página de login
11     header("Location: login.php");
12 }
13 ?>
```

Esse arquivo é o responsável para verificar se o usuário tem uma sessão aberta. Caso tenha, ele continua navegando na página. Caso contrário, ele será redirecionado para a página de login.

Arquivo index.php

Figura 75 - Arquivo index.php

```
1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6
7 require_once("validaSessao.php"); // chama o arquivo para validar a sessão
8 ?>
9 <html>
10    <head>
11        <title>IFES - Bem vindo!</title>
12        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
13    </head>
14    <body>
15        <table width="400" border="1" align="center" cellpadding="5" cellspacing="0">
16            <tr>
17                <th bgcolor="#E5E5E5" scope="col">Seja bem vindo!</th>
18            </tr>
19            <tr>
20                <td><ul>
21                    <li><a href="addUsuario.php">Cadastrar Usuário</a></li>
22                    <li><a href="usuarios.php">Exibir Usuários</a></li>
23                    <li><a href="sair.php">Sair</a></li>
24                </ul></td>
25            </tr>
26        </table>
27    </body>
28 </html>
```

Veja que, em todas as páginas que o usuário necessita estar logado para poder acessar, nós chamamos o arquivo de validaSessao.php. Assim, só poderá acessar essas páginas quem estiver logado.

Arquivo addUsuario.php

Figura 76 - Arquivo addUsuario.php

```
1 <?php
2     ////////////////// DADOS DO DESENVOLVEDOR //////////////////
3     # Nome: Flávio Izzo
4     # Email: flavio@flavioizzo.com
5     /////////////////////////////////
6
7     require_once("validaSessao.php"); // chama o arquivo para validar a sessão
8 ?>
9 <html>
10    <head>
11        <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
12        <title>IFES - Cadastrar Usuário</title>
13    </head>
14    <body>
15        <form method="post" action="procAddUsuario.php" name="Form1">
16            <table width="500" border="1" align="center" cellpadding="5" cellspacing="0">
17                <tr>
18                    <td colspan="2" align="center" bgcolor="#E9E9E9">Cadastro de Usuário</td>
19                </tr>
20                <tr>
21                    <td>Nome</td>
22                    <td><input name="nome" type="text" size="60"></td>
23                </tr>
24                <tr>
25                    <td>Login</td>
26                    <td><input name="login" type="text"></td>
27                </tr>
28                <tr>
29                    <td>Senha</td>
30                    <td><input name="senha" type="password"></td>
31                </tr>
32                <tr>
33                    <td>Situação</td>
34                    <td>
35                        <select name="situacao">
36                            <option value="1" selected>Ativo</option>
37                            <option value="0">Inativo</option>
38                        </select>
39                    </td>
40                </tr>
41                <tr>
42                    <td colspan="2" align="center">
43                        <input name="Submit" value="Inserir Dados" type="submit">
44                    </td>
45                </tr>
46            </table>
47        </form>
48    </body>
49 </html>
```

Essa é uma página de login comum para efetuar os cadastros dos usuários. Na linha 15, podemos observar que a página chamada quando o formulário for submetido será a procAddUsuario.php.

Arquivo procAddUsuario.php

Figura 77 - Arquivo procAddUsuario.php

```
1  <?php
2  ##### DADOS DO DESENVOLVEDOR #####
3  # Nome: Flávio Izo
4  # Email: flavio@flavioizo.com
5  #####
6  // valida a sessão
7  require_once("validaSessao.php");
8  ?>
9  <html>
10 <head>
11   <title>Processando cadastro do usuário</title>
12   <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
13   <META HTTP-EQUIV="Refresh" CONTENT="3;URL=usuarios.php">
14 </head>
15 <body>
16   <?php
17   require("conexao.php"); //chama o arquivo de conexão ao BD
18
19   /*"pega" os dados digitados no form, através do método POST.*/
20   $nome = $_POST['nome'];
21   $login = $_POST['login'];
22   $senha = md5($_POST['senha']); // md5 é uma função que criptografa a senha
23   $situacao = $_POST['situacao'];
24
25   /*Inserindo os dados na Tabela "dados" através de comandos MySQL.*/
26   $sql = "INSERT INTO usuarios (nome, login, senha, situacao)
27         VALUES ('$nome', '$login', '$senha', '$situacao')";
28
29   mysql_query($sql) or die("Não foi possível inserir os dados");
30
31   echo "<br /><center>Dados inseridos corretamente!</center>";
32   echo "Redirecionando em 3 segundos..."; 
33   ?>
34 </body>
35 </html>
```

Esta é a página que recebe os dados enviados pelo formulário.

Na linha 13, utilizamos o recurso de refresh fornecido pelo html. Esse recurso atualiza ou redireciona a página depois de certo tempo. No exemplo, a página será redirecionada para usuarios.php após 3 segundos.

Na linha 22, criptografamos a senha do usuário para gravar no banco de dados.

Arquivo usuarios.php

Figura 78 - Arquivo usuarios.php

```

1: <?php
2: //***** DADOS DO DESENVOLVEDOR *****
3: # Nome: Flávio Izo
4: # Email: flavio@flavioizo.com
5: // valida a sessão
6: // valida a sessão
7: require_once("validaSessao.php");
8?>
9<html>
10<head>
11    <title>ITES - Exibindo Usuários</title>
12    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
13</head>
14<body>

15    <?php
16    require("conexao.php"); //chama o arquivo de conexão com o BD
17
18    $sql = "SELECT * FROM usuarios";
19    $sql_exec = mysql_query($sql);
20?
21    <table width="500" border="0" align="center" cellpadding="3" cellspacing="1">
22        <tr>
23            <th colspan="4" bgcolor="#E9E9E9" scope="col">MENU</th>
24        </tr>
25        <tr>
26            <th bgcolor="#BFD9FF" scope="col"><a href="index.php">Principal</a></th>
27            <th bgcolor="#BFD9FF" scope="col"><a href="addUsuario.php">Cadastrar Usuário</a></th>
28            <th bgcolor="#BFD9FF" scope="col"><a href="usuarios.php">Exibir Usuário</a></th>
29            <th bgcolor="#BFD9FF" scope="col"><a href="sair.php">Sair</a></th>
30        </tr>
31    </table>
32
33    <p>&ampnbsp</p>
34    <table width="600" border="1" align="center" cellpadding="0" cellspacing="0">
35        <tr>
36            <th colspan="7" bgcolor="#E9E9E9">Dados cadastrados</th>
37        </tr>
38        <tr>
39            <th> Código </th>
40            <th> Nome </th>
41            <th> Login </th>
42            <th> Senha </th>
43            <th> Situação </th>
44            <th bgcolor="#BFD9FF"> AÇÕES </th>
45        </tr>
46    </table>
47    <?php
48    //loop para exibir os dados da tabela
49    while ($linha = mysql_fetch_array($sql_exec)) {
50        $idUser = $linha["idUser"];
51        $nome = $linha["nome"];
52        $login = $linha["login"];
53        $senha = $linha["senha"];
54        $situacao = $linha["situacao"];
55
56        //mostra os dados na tela
57   ?>
58        <tr>
59            <td align="center"><?php echo $idUser; ?></td>
60            <td align="center"><?php echo $nome; ?></td>
61            <td align="center"><?php echo $login; ?></td>
62            <td><?php echo $senha; ?></td>
63            <td align="center"><?php if ($situacao == 1) {
64                echo "Ativo";
65            } else {
66                echo "Inativo";
67            } ?></td>
68            <td align="center" valign="middle" bgcolor="#BFD9FF">
69                <a href="altUsuario.php?id=<?php echo $idUser; ?>">
70                    
71                </a>&ampnbsp&ampnbsp
72                <a href="delUsuario.php?id=<?php echo $idUser; ?>">
73                    
74                </a>
75            </td>
76        </tr>
77        <?php
78    }
79    ?>
80    </table>
81</body>
82</html>

```

Esta página é responsável por listar todos os usuários cadastrados no BD. Nas linhas 69 e 72, quando for chamado o link de alteração ou de deleção, temos que passar como parâmetro o id do usuário que será alterado ou excluído.

Arquivo altUsuario.php

Figura 79 - Arquivo altUsuario.php

```
1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6
7 require_once("validaSessao.php"); // chama o arquivo para validar a sessão
8 ?>
9 <html>
10 <head>
11   <meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
12   <title>IPES - Alterar Usuário</title>
13 </head>
14 <body>
15   <?php
16     require("conexao.php"); //chama o arquivo de conexão ao BD
17     $idUser = $_GET["id"];
18     $sql = "SELECT * FROM usuarios WHERE idUser='$idUser'";
19     $sql_exec = mysql_query($sql); //verifica o limite da tabela
20
21     $nome = mysql_result($sql_exec,0,1);
22     $login = mysql_result($sql_exec,0,2);
23     $senha = mysql_result($sql_exec,0,3);
24     $situacao = mysql_result($sql_exec,0,4);
25   ?>
26   <form method="post" action="procAltUsuario.php" name="Form1">
27     <table width="500" border="1" align="center" cellpadding="5" cellspacing="0">
28       <tr>
29         <td colspan="2" align="center" bgcolor="#E9E9E9">Alterar Usuário</td>
30
31       </tr>
32       <tr>
33         <td>Nome</td>
34         <td><input name="name" type="text" size="60" value="<?php echo $nome;?>">
35           <input name="idUser" type="hidden" value="<?php echo $idUser;?>">
36         </td>
37       </tr>
38       <tr>
39         <td>Login</td>
40         <td><input name="login" type="text" value="<?php echo $login;?>"></td>
41       </tr>
42       <tr>
43         <td>Senha</td>
44         <td><input name="senha" type="password"></td>
45       </tr>
46       <tr>
47         <td>Situação</td>
48         <td>
49           <select name="situacao">
50             <option value="1" ><?php if ($situacao == 1) {
51               echo "selected";
52             ?>>>Ativo</option>
53             <option value="0" ><?php if ($situacao == 0) {
54               echo "selected";
55             ?>>>Inativo</option>
56           </select>
57
58         </td>
59       </tr>
60       <tr>
61         <td colspan="2" align="center">
62           <input name="Submit" value="Atualizar Dados" type="submit">
63         </td>
64       </tr>
65     </table>
66   </form>
67 </html>
```

Nesta página, temos que dar destaque para a linha 35, haja vista que estamos colocando o código do usuário como campo oculto (*hidden*), pois o usuário não poderá alterá-lo. No entanto, é necessário ter esse campo, pois quando o formulário for enviado, vamos atualizar os dados relativos a esse id.

Arquivo procAltUsuario.php

Figura 80 - Arquivo procAltUsuario.php

```

1  <?php
2      ##### DADOS DO DESENVOLVEDOR #####
3      # Nome: Flávio Izo
4      # Email: flavio@flavioizo.com
5      #####
6      // valida a sessão
7      require_once("validaSessao.php");
8      ?>
9      <html>
10     <head>
11         <title>Processando alteração do usuário</title>
12         <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
13         <META HTTP-EQUIV="Refresh" CONTENT="3;URL=usuarios.php">
14     </head>
15     <body>
16         <?php
17         require("conexao.php"); //chama o arquivo de conexão ao BD
18
19         /*"pega" os dados digitados no form, através do método POST.*/
20         $idUser = $_POST['idUser'];
21         $nome = $_POST['nome'];
22         $login = $_POST['login'];
23
24         $sql = "SELECT senha FROM usuarios WHERE idUser='$idUser'";
25         $sql_exec = mysql_query($sql);
26         // pega a senha para o caso do usuário não ter alterado-a
27         $senha = mysql_result($sql_exec,0,0);
28
29         if (!empty($_POST['senha'])){
30             $senha = md5($_POST['senha']); // md5 é uma função que criptografa a senha
31         }
32         $situacao = $_POST['situacao'];
33
34         /*atualizando os dados na Tabela "usuarios" através de comandos MySQL.*/
35         $sql = "UPDATE usuarios SET nome='$nome', login='$login',
36                 senha='$senha', situacao='$situacao' WHERE idUser='$idUser'";
37         mysql_query($sql) or die("Não foi possível atualizar os dados");
38
39         echo "<br /><center>Dados atualizados corretamente!</center>";
40         echo "Redirecionando em 3 segundos...";
41     ?>
42     </body>
43 </html>
```

Arquivo delUsuario.php

Figura 81 - Arquivo delUsuario.php

```
1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6 // valida a sessão
7 require_once("validaSessao.php");
8 ?
9 <html>
10 <head>
11     <title>Processando alteração do usuário</title>
12     <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
13     <META HTTP-EQUIV="Refresh" CONTENT="3;URL=usuarios.php">
14 </head>
15 <body>
16     <?php
17     require("conexao.php"); //chama o arquivo de conexão ao BD
18
19     /*"pega" os dados digitados no form, através do método POST.*/
20     $idUser = $_GET['id'];
21
22     /*apagando os dados na Tabela "usuarios" através de comandos MySQL.*/
23     $sql = "DELETE FROM usuarios WHERE idUser='$idUser'";
24     mysql_query($sql) or die("Não foi possível apagar os dados");
25
26     echo "<br /><center>Dados apagados corretamente!</center>";
27     echo "Redirecionando em 3 segundos..."; 
28     ?
29     </body>
30 </html>
```

Arquivo sair.php

Figura 82 - Arquivo sair.php

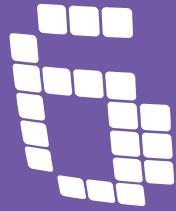
```
1 <?php
2 ##### DADOS DO DESENVOLVEDOR #####
3 # Nome: Flávio Izo
4 # Email: flavio@flavioizo.com
5 #####
6
7 session_start();
8 session_destroy(); // mata as sessões existentes
9
10 //redirecionamos para a página de login
11 header("Location: login.php");
12 ?>
```

Esse arquivo vai destruir a sessão e redirecionar o usuário para a página login.



Anotações

Handwriting practice lines. There are 21 horizontal lines for practice, each consisting of a solid top line, a dashed midline, and a solid bottom line.



CLASSES E OBJETOS

Olá!

Neste sexto capítulo faremos uma pequena abordagem sobre classes e objetos.

O objetivo deste capítulo é que você conheça que o PHP também passou a ter suporte à OO.

As informações disponíveis neste capítulo fornecerão um suporte para que, em projeto futuros, você possa desenvolver seus projetos em OO.

Bons estudos!

Fala Professor

Vogue risus al
e velit at tellus.
massa portitor
sectetur magna.

Apesar de não ser nativo à OO, a partir da versão 5 o PHP também passou a aderir a esse conceito: orientação a objetos.

Infelizmente não temos como abordar tudo que a OO tem a oferecer, porém vamos citar um exemplo simples para que você possa aperfeiçoá-lo sempre que necessário.

O PHP OO utiliza os mesmos princípios da OO: Classe, objetos, Herança, polimorfismo etc. Então vamos ao nosso exemplo:

Crie a classe Produto e salve como Produto.class.php

Figura 83 - Classe Produto (Produto.class.php)

```

1 <?php
2
3 class Produto{
4
5     public $Codigo;
6     public $Nome;
7     public $Preco;
8     public $Distribuidor;
9
10    public function __construct($Codigo, $Nome, $Preco, $Fornecedor){
11        $this->Codigo = $Codigo;
12        $this->Nome = $Nome;
13        $this->Preco = $Preco;
14        $this->Distribuidor = $Fornecedor;
15    }
16
17 }
18 ?>
```

Crie a classe Distribuidor e salve como Distribuidor.class.php

Figura 84 - Classe Distribuidor (Distribuidor.class.php)

```
1 <?php
2
3 class Distribuidor{
4
5     public $Codigo;
6     public $RazaoSocial;
7     public $Endereco;
8     public $Telefone;
9
10    public function __construct($Codigo, $RazaoSocial, $Endereco, $Telefone) {
11        $this->Codigo = $Codigo;
12        $this->RazaoSocial = $RazaoSocial;
13        $this->Endereco = $Endereco;
14        $this->Telefone = $Telefone;
15    }
16
17 }
18 ?>
```

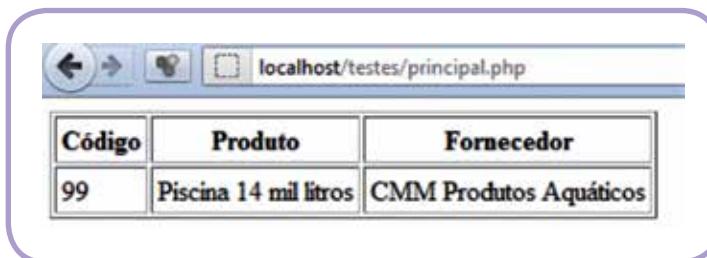
Crie, agora, o arquivo principal.php que funcionará como o código que vai chamar as classes.

Figura 85 - Arquivo principal principal.php

```
1 <?php
2 require_once("Distribuidor.class.php");
3 require_once("Produto.class.php");
4 $distribuidor = new Distribuidor(1, "CMM Produtos Aquáticos",
5                                 "Rua da Saúde, 27", "9999-9999");
6
7 $produto = new Produto(99, "Piscina 14 mil litros", 11000.00, $distribuidor );
8
9 ?>
10 <html>
11   <head>
12     <title>Meu formulário</title>
13   </head>
14   <body>
15   <table border="1" cellpadding="3">
16     <tr>
17       <th> Código </th>
18       <th> Produto </th>
19       <th> Fornecedor </th>
20     </tr>
21     <tr>
22       <td><?php echo $produto->Codigo; ?></td>
23       <td><?php echo $produto->Nome; ?></td>
24       <td><?php echo $produto->Distribuidor->RazaoSocial; ?></td>
25     </tr>
26   </table>
27   </body>
28 </html>
```

Ao ser interpretado, a página principal.php vai gerar a seguinte saída:

Figura 86 - Arquivo principal.php interpretado pelo servidor web



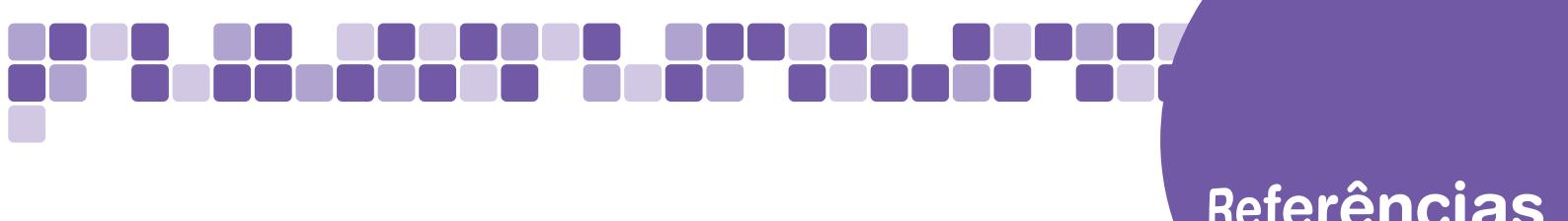
| Código | Produto | Fornecedor |
|--------|-----------------------|------------------------|
| 99 | Piscina 14 mil litros | CMM Produtos Aquáticos |

Assim, encerramos esta apostila. Espero que tenham gostado!



Anotações

Handwriting practice lines for notes.



Referências

CONVERSE, Tim; PARK, Joyce. **PHP 4 - A Bíblia.** 2ª Edição. São Paulo: Campus, 2003.

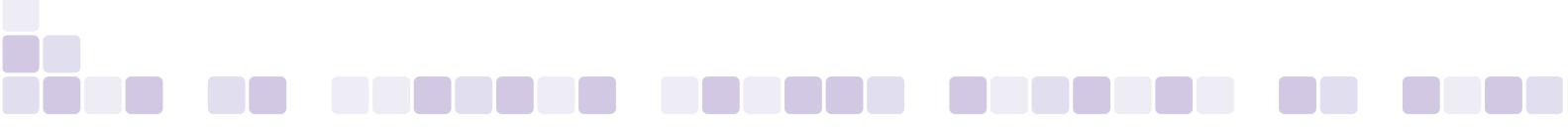
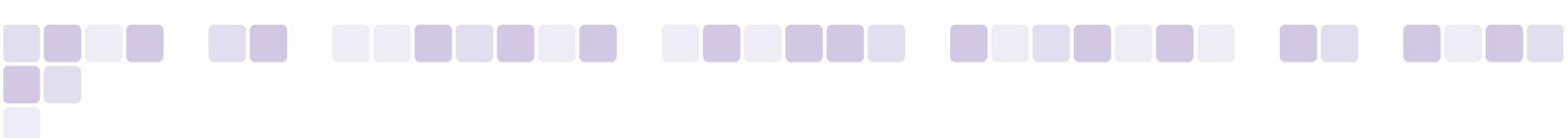
FREEMAN; Elisabeth; FREEMAN, Eric. **Use a Cabeça HTML com CSS e XHTML.** Rio de Janeiro: Alta Books, 2008. GOLÇALVES, Edson. **Desenvolvendo Aplicações Web com NetBeans IDE 6.** São Paulo: Ciência Moderna, 2008.

MANUAL DO PHP. Disponível em <http://www.php.net/manual/pt_BR/>

MARCONDES, Christian. **HTML 4.0 FUNDAMENTAL – A BASE DA PROGRAMAÇÃO PARA WEB.** São Paulo: ÉRICA, 2005.

THONSON, Laura; WELLING, Luck. **PHP e MYSQL: Desenvolvimento Web.** 3ª Edição. São Paulo: Campus, 2005.

W3C Brasil. Disponível em: <<http://www.w3c.br>>



A UFRR ESTÁ AQUI ROMPENDO FRONTEIRAS

