Introdução a programação em PHP 5 utilizando o Wamp 5

Conteúdo prático para iniciantes

- Introdução a PHP
- Resumo sobre Orientação a Objetos
- Breve resumo sobre MVC

Ilton Barbosa - jittobr@gmail.com
Fórum: http://iltonbarbosa.netfreehost.com/

Sumário

Introdução

Primeiramente parabenizo-lhe pelo seu interesse em buscar conhecimentos por sua própria iniciativa, com o fim de aperfeiçoamento profissional, agregando valor ao seu currículo e consequentemente melhor se posicionando no mercado de trabalho.

Nesta apostila não vou me ater ao uso corrente de termos muito técnicos e procurarei utilizar uma linguagem a mais simples possível, de forma que o aprendizado ocorra com naturalidade e seja de fácil compreensão para qualquer pessoa.

Ainda assim aconselho o aprendizado de termos técnicos, pois saber empregá-los corretamente lhe garantirá maior credibilidade no mercado. É a forma de expressar profissionalmente seus conhecimentos, portanto considero esse fator indispensável quando no caso de uma entrevista de trabalho.

Esta apostila apresenta o assunto de uma forma bem suscinta, evitando textos longos e explicações muito minunciosas, com o objetivo de proporcionar uma leitura agradável e pouco cansativa.

Estudando esta apostila com dedicação, certamente voce terá condições de dar continuidade aos estudos de programação em PHP, por sua própria conta. O meu propósito é orientá-lo para o estudo de PHP.

Pré-requisitos para o estudo desta apostila

Como pré-requisito para este estudo é necessário ter um conhecimento mínimo de **HTML**, noções de **banco de dados** com o uso de SQL e noções de **lógica de programação**. Estes conhecimentos não são difíceis de se ter, na internet é possível encontrar muitas apostilas sobre esses assuntos.

Se lhe falta um destes conhecimentos, este estudo será um pouco mais difícil, mas mesmo assim, não é impossível que se aprenda alguma coisa aqui.

O PHP

Se voce já sabe criar páginas para a internet utilizando HTML, saiba que com PHP elas poderão ficar ainda melhores, por possibilitar que seu site tenha funcionalidades como cadastro de usuários, forum, blogs, envio de mensagens e tudo o que se refere a interação com o usuário do site.

Você pode encontrar na internet muito mais informações sobre o que é o PHP, resumidamente trata-se de uma linguagem de programção que, ao ser intepretada por um servidor WEB, o código PHP se transforma em HTML. Os comandos em PHP servem para buscar informações no seu banco de dados e enviá-las ao servidor, para que este as converta em HTML e possam ser lidas pelo seu browser de internet.

Existem muitas outras linguagens de programação que fazem a mesma coisa que o PHP, como por exemplo, o JAVA, o dotNet, o Python, Ruby, ASP e por aí vai. Dentre elas, na minha opinião, a mais fácil de se programar atualmente é o PHP.

O WAMP

Para trabalhar com PHP é necessário ter instalado na sua máquina um servidor que irá interpretar o código PHP e gerar um código HTML, como também um banco de dados para guardar informações. O servidor que utilizaremos é o APACHE e o gerenciador de banco de dados é o MySQL. Não vou me ater aqui aos conceitos sobre o que é o Apache e o que é o MySQL. Estas informações estão na internet aos montes, pesquise sobre elas para que tenha um maior conhecimento a respeito.

O Wamp é um aplicativo gratuito para Windows que instala e configura automaticamente na sua máquina o servidor Apache, o Banco de dados MySQL e o PHP, além de disponibilizar ferramentas que facilitam a criação do banco de dados com suas tabelas e relacionamentos. As duas ferramentas para trabalhar com banco de dados, disponibilizadas pelo Wamp são o PhpMyAdmin e o SqlLiteManager, nesta apostila trabalharemos apenas com o PhpMyAdmin.

O primeiro passo para praticar os exemplos desta apostila na sua máquina é **baixar da internet a última versão do Wamp** e instalar no seu computador. A instalação é muito simples, bastando clicar em "next" nas telas de instalação. Para esta apostila utilizei o Wamp5 versão 1.7.3.

Obs.: Wamp é a abreviação de **W**indows, **A**pache, **M**ySql e **P**hp. Para Linux temos o Lamp.

Na internet tem um forum muito bom para estudo e para enviar suas dúvidas.
Acesse: http://www.phpbrasil.com/phorum/index.php

PRIMEIROS PASSOS: Testando o servidor Web local

Ao instalar o Wamp, observe que no canto inferior direito da sua tela aparecerá um ícone semelhante a este da figura ao lado. Ele indica que o Wamp está ativo.

Quando o Wamp está ativo sua máquina se torna um servidor Web para acesso local. Faça o seguinte teste:

- Clique no ícone do Wamp e selecione a opção **localhost**, se aparecer uma tela conforme a imagem abaixo, quer dizer que o seu Wamp está instalado corretamente.

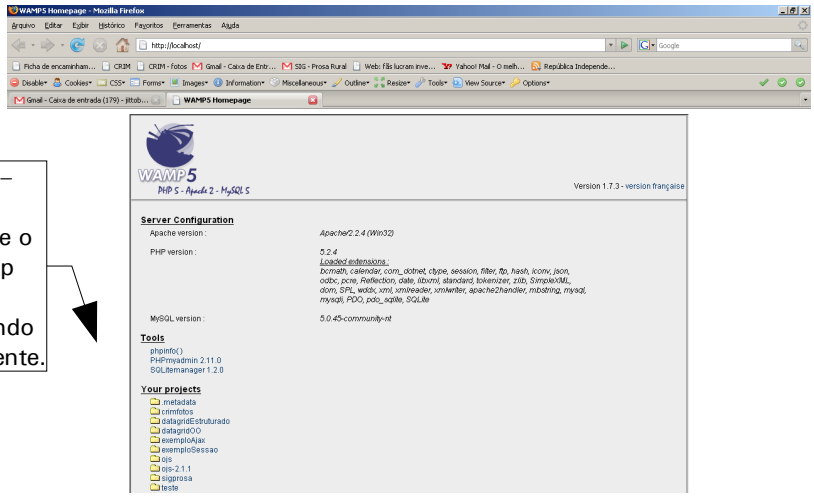


Figura 1 – Esta tela indica que o seu Wamp está funcionando corretamente.

Observe que, basta abrir o browser e digitar http://localhost no campo de endereços.

Testando o PhpMyAdmin

O PhpMyAdmin é a ferramenta que utilizaremos para facilitar a criação e manutenção do nosso banco de dados. Para testar se ele está funcionando corretamente, faça o seguinte:

- Clique no ícone do Wamp e selecione a opção "**phpMyAdmin**". Deverá aparecer uma tela conforme a imagem abaixo:

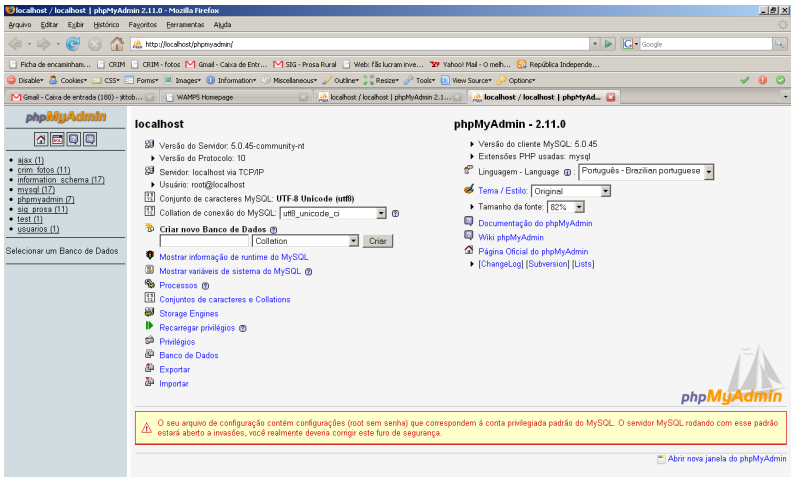


Figura 2 – Esta tela indica que o phpMyAdmin está funcionando corretamente

Depois que estiver mais familiarizado com o Wamp, dê uma olhada nos outros recursos que estão no seu menu, como por exemplo o "PHP settings" e o "Config files", que em alguns casos precisam ser utilizados para eventuais alterações nas configurações do PHP ou do Apache, mas isso não será necessário nos ensinamentos desta apostila. Por enquanto não faça alterações nestes arquivos para evitar problemas ao praticar os exercícios aqui propostos.

Não pretendo abordar aqui conceitos básicos de programação e sua lógica. Estes são pré-requisitos para este estudo portanto vamos partir logo para a prática. Sobre PHP e MySql, explicarei apenas os comandos que estiverem nos exemplos desta apostila. Recomendo ao leitor buscar ler mais a respeito.

Na minha opinião não é necessário ler um livro sobre todos os comandos do PHP, aprenda aos poucos, estude-os à medida que for precisando utilizá-los. O que poderá acontecer é você, só mais tarde, descobrir que existem formas mais fáceis de se fazer alguns procedimentos que voce já havia aprendido. Se quiser evitar isso, então faça apenas uma leitura rápida dos comandos mais utilizados no PHP, só para saber que eles existem, sem precisar ter que ficar decorando a forma de como utilizá-los. Quando precisar utilizar um deles, basta pesquisar no livro ou internet para saber como utilizá-lo. Depois, com o uso constante destes comando, voce vai aos poucos aprendendo sobre eles.

Editor para escrever os códigos PHP

Para escrever seus códigos em PHP, pode-se utilizar até mesmo o NotePad do Windows, no entanto existem softwares que facilitam bastante a programação, como por exemplo, o DreamWeaver, o EasyEclipse para PHP, o NetBeans para PHP e muitos outros. Pesquise no Google sobre editores para PHP e verá que a lista é grande.

Voce pode utilizar qualquer um deles para praticar os exercícios aqui propostos. Se eu direcionasse o estudo para um destes editores, teria que explicar como utilizá-lo, o que implicaria em aumento no número de páginas da apostila. A intenção é que ela seja bem resumida e didaticamente fácil.

Aconselho que utilizem um editor que pode lhe auxiliar na programação. O EasyEclipse e o NetBeans para PHP são gratuitos, mas exige um pouco de conhecimento sobre a sua utilização. Vale à pena estudá-los, pois são muito utilizados por profissionais da área. O DreamWeaver é a melhor ferramenta para se trabalhar com HTML, mas não é gratuito. Não ajuda muito no código PHP, tem alguns recursos como destacar em vermelho os comandos em PHP que voce digitar e também automatiza a criação de alguns códigos em PHP, só que não são muito limpos. Atualmente utilizo o EasyEclipse para criar os códigos e o DreamWeaver só para a criação das telas em html.

Se for utilizar o NotePad do Windows, tome cuidado na hora de salvar os arquivos, pois ele costuma colocar ".txt" no final do nome dos arquivos.

Iniciando o desenvolvimento de um sistema básico

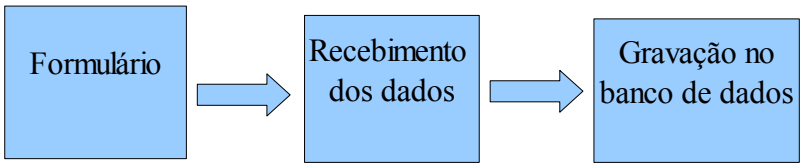
Vou mostrar aqui, um DataGrid bem simples. DataGrid é o conjunto das funções para Consultar, Gravar, Alterar e Excluir informações de cadastro de um sistema. Aprendendo isso, você já saberá o básico de qualquer sistema.

Vamos criar telas e formulários bem simples para o nosso sistema, porque o objetivo aqui também não é ensinar layout (webdesign). E se entrássemos nestes detalhes estaríamos desviando o foco principal desta apostila e, consequentemente ela ficaria com muito mais páginas.

Nosso sistema terá um formulário para cadastro, outro para alteração e uma tela de consulta. Na tela de alteração já implementaremos o recurso de exclusão.

Teremos arquivos para cada uma das telas e arquivos para receber os dados dos formulários. Esclarecendo melhor, um sistema de cadastro funciona da seguinte forma: O usuário preenche as informações do formulário e, ao clicar no botão para gravar, as informações fornecidas pelo usuário são transferidas para um outro arquivo, que por sua vez, executará a gravação das mesmas no banco de dados. Se tudo correr bem, o sistema deverá exibir mensagem "Gravação efetuada com sucesso!", caso não consiga gravar exibe mensagem de erro.

A figura abaixo demonstra a fluxo básico do sistema:



Os formulários de cadastro e alteração terão códigos em html e alguns comandos PHP, já o arquivo que receberá os dados do formulário terá muito PHP e pouco html.

O correto é separmarmos o máximo possível os códigos em html dos comandos PHP, mas isso poderá ser aprendido depois com mais estudo. Quem sabe numa próxima apostila.

- Primeira tarefa

Na pasta onde o Wamp foi instalado, observe que existe uma subpasta chamada "www", é nesta subpasta que deveremos criar os sistemas a serem testados na nossa máquina local. Esta pasta pode ser acessada clicando no ícone do Wamp (no canto inferior direito da tela – já falei sobre este ícone na página 5) e selecionando a opção "**www directory**".

Crie uma pasta dentro da pasta www, com o nome de **dgEstruturado**. Este é um nome qualquer que sugeri, "dg" significa DataGrid e "Estruturado" porque inicialmente não utilizaremos programação orientada a objetos. Posteriormente, dentro desta pasta voce deverá gravar todos os arquivos do seu sistema para que possam ser lidos pelo Apache e convertidos para html.

Dentro da pasta **dgestruturado**, crie um arquivo chamado **cadastro_usuario.htm**. Este arquivo deverá ter um formulário com os seguintes campos:

- nome_usuario (size 50);
- e_mail_usuario (size 50);
- telefone_usuario (size 30);

Além do botão para submeter os dados.

O **código do formulário** em html ficará basicamente assim:

(lembre-se que utilizando o DreamWeaver não é necessário digitar o código, basta criar fazendo uso dos recursos de formulário que ele já disponibiliza)

```
<html>
  <body>
    <p>Cadastro de Uusários</p>
    <form name="cadastro de usuario" method="post" action="cadastro_usuario_exe.php">
      <table>
        <tr>
          <td>
            Nome:</td><td><input name="nome_usuario" type="text" size="50" maxlenght="50">
          </td>
        </tr>
        <tr>
          <td>E-mail:</td>
          <td><input name="e_mail_usuario" type="text" size="50" maxlenght="50"></td>
        </tr>
        <tr>
          <td>Telefone:</td>
          <td><input name="telefone_usuario" type="text" size="30" maxlenght="30"></td>
        </tr>
        <tr>
          <td colspan=2 align="center"><input type="submit" name="Submit" value="Enviar"></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

Abra o browser de internet e digite
http://localhost/dgestruturado/cadastro_usuario.htm

Na tela deverá aparecer o formulário que voce acabou de criar.

Esteja à vontade para enfeitar o seu formulário.

O nome em vermelho "**cadastro_usuario_exe.php**" é o nome do arquivo que receberá os dados preenchidos neste formulário (*coloquei em vermelho só para destacar*)

Mas como este arquivo receberá os dados?

Escrevendo seu primeiro código em PHP

Crie um arquivo com o nome "**cadastro_usuario_exe.php**" e digite o código abaixo: *(estes arquivos poderiam ter qualquer nome, eu escolhi estes)*

```
1  <?php
2  $nome_usuario = $_POST['nome_usuario'];
3  $e_mail_usuario = $_POST['e_mail_usuario'];
4  $telefone_usuario = $_POST['telefone_usuario'];
5
6  echo "<P>Nome do usuário: ".$nome_usuario."<BR>";
7  echo "E-mail: ".$e_mail_usuario."<BR>";
8  echo "Telefone: ".$telefone_usuario."</P>";
9  ?>
```

Este é o nome do campo no formulário

Observe que o nome do campo no formulário é o mesmo nome da variável. Só por questão de organização.

- Explicando o código:

Todo arquivo php deve iniciar com o comando "<?php" e fechar com o comando "?>", conforme está na linha 1 e linha 9.

Cada linha de comando em PHP deve ser finalizada com ponto-e-vírgula (;).

Da linha 2 a 4 criei variáveis para receber os dados do formulário. Observe que o nome de toda variável em php deve iniciar-se com o símbolo "$".

Fique atento aos nomes que for utilizar para suas variáveis.

Conforme o exemplo dos dois códigos acima, observe que o nome da variável no segundo código é o mesmo nome utilizado para nomear o campo do formulário (no primeiro código). Para facilitar o trabalho de um programador, o ideal é que o nome das variáveis sejam os mesmos nomes utilizados para nomear os campos do formulário que voce criar. E também que estes nomes sejam os mesmos empregados nos campos das tabelas do seu banco de dados, que veremos mais adiante.

Para receber os valores vindos do formulário utilizei o comando "$_POST". Observe que na quarta linha do código html do formulário eu defini que o método de envio dos dados do formulário é o método "POST", portanto utilizei o comando "$_POST" no php para pegar estes dados, conforme está nas linhas 2, 3 e 4.

- Existem duas maneiras para enviar os dados: método POST e método GET.

Qual a diferença entre os métodos POST e GET?

A diferença está no seguinte: enviando os dados pelo método GET, quando o usuário clicar no botão para envio dos dados, estas informações irão aparecer naquele campo do browser em que se digita os endereços de internet; utilizando o método POST eles não aparecerão lá. Entendeu?

Da sexta à oitava linha utilizei o comando "echo" para exibir os dados que foram preenchidos no formulário. Observe que nestas linhas há comandos html no meio de comandos PHP. Estes comandos em html estão entre aspas. **Qualquer texto a ser mostrado na tela do usuário, utilizando PHP, deverá estar entre aspas**, com exceção das variáveis.

Se quiser exibir na tela um texto e o conteúdo de uma variável utilize a **concatenação**. Para concatenar texto e variáveis utiliza-se o ponto (.). Observe o ponto sendo utilizado nas linhas 6, 7 e 8, separando textos e variáveis (isso é concatenação).

Agora teste este código na sua máquina local, utilizando o Wamp. Abra o browser de internet e digite http://localhost/dgestruturado/cadastro_usuario.htm, preencha o formulário e clique no botão "**Enviar**".

Se conseguir fazer funcionar nos conformes, então pode continuar os estudos desta apostila. Se não conseguir, mande-me mensagem para tirar suas dúvidas
(jittobr@gmail.com) ou, para fazer parte de um ambiente onde possamos nos interagir melhor, cadastre-se na comunidade da **Tecnociencia.org**, acessando o seguinte endereço: **http://tecnociencia.inf.br/tecnico/iltonBarbosa**

Uma outra opção é cadastrar-se no forum que criei só para tratar de assuntos referentes a esta apostila e envie suas dúvidas:

http://iltonbarbosa.netfreehost.com/

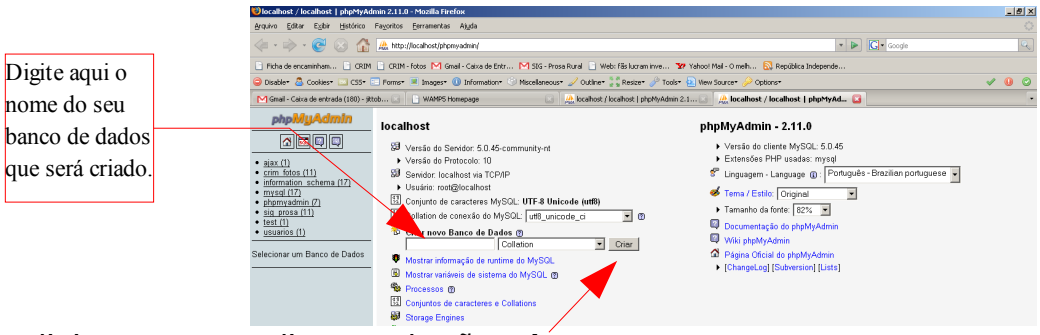
Criando um Banco de dados no PhpMyAdmin

Vamos agora criar o banco de dados que armazenará os dados preenchidos no formulário.

Acesse o PhpMyAdmin conforme está na página 5. Na tela que se apresenta, digite o nome do banco no campo indicado abaixo:

O nome do nosso banco será o mesmo nome que utilizamos para o nosso sistema, por questão de organização. E que por este mesmo motivo, é o nome da pasta que criamos no diretório WWW.

O nome é **dgEstruturado**.



Após digitar o nome, clique no botão **criar**.

Prontinho, o seu banco está criado. Agora vamos criar as tabelas do banco.

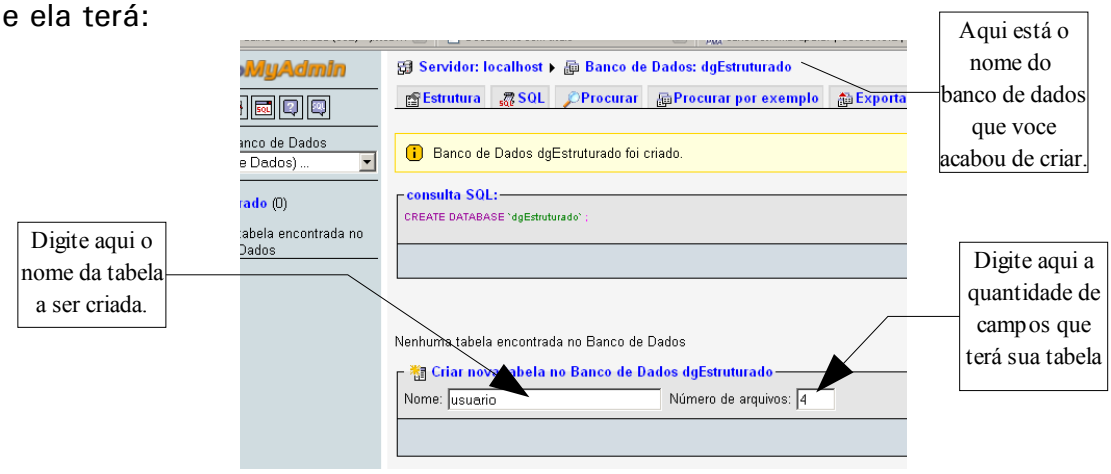
Inicialmente nosso banco terá apenas uma única tabela, a qual chamaremos de "usuario" (sem acentuação).

Esta tabela terá 4 campos, conforme os campos que criamos no formulário e mais o campo "id_usuario" que é o **número de registro único** do usuário no banco. Sendo assim os campos serão os seguintes:

- id_usuario: int(4)
- nome_usuario: varchar(50);
- e_mail_usuario: varchar(50);
- telefone_usuario: varchar(30);

Os campos "varchar(50)" indicam que serão armazenados textos com até 50 caracteres.

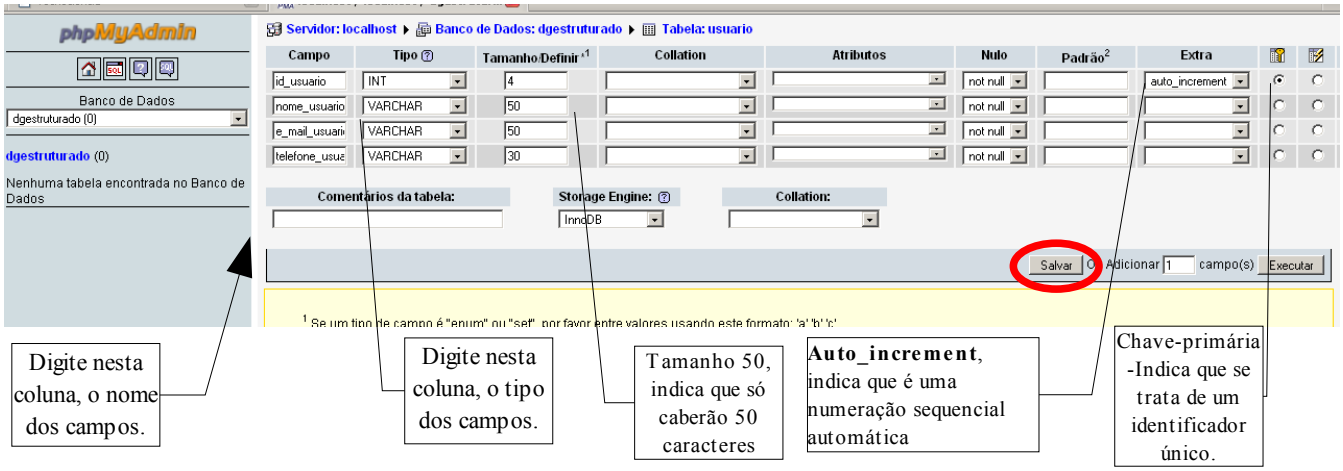
Conforme a tela abaixo digite o nome da tabela e a quantidade de campos que ela terá:



Clique o botão "executar" que está à sua direita.

Na tela seguinte crie os campos do seu banco, conforme a figura abaixo, obedecendo os nomes dos campos corretos:

id_usuario,nome_usuario,e_mail_usuario e telefone_usuario.

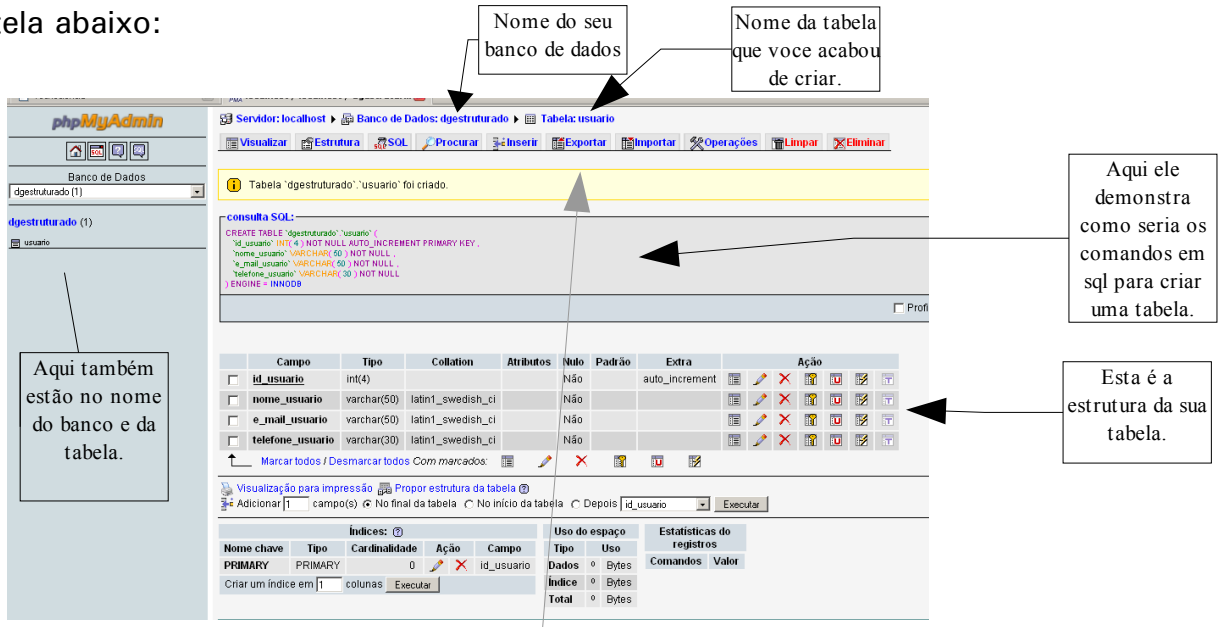


Observe que o campo id_usuario é **chave primária** e **auto_increment**. Se voce esqueceu de definir estes detalhes, acesse o recurso de alteração do campo, clicando no ícone em forma de lápis que está na mesma linha do nome do campo.

Depois de definir tudo, clique no botão **Salvar**.

Se não tem muita experiência com SQL (MySQL), recomendo que baixe da internet algumas apostilas sobre esse assunto e estude-as, isso é de fundamental importância para quem deseja ser um bom programador.

Depois que você clicou no botão "Salvar", o PhpMyAdmin, lhe apresenta a tela abaixo:



Agora vamos verificar se voce fez tudo corretamente.

- Clique no nome da tabela, que está aparecendo no seu lado esquerdo.
- Clique, em seguida, na aba "Exportar".
- E então clique no botão "executar" que aparece no canto inferior direito.

Na tela seguinte, será mostrado o código em SQL de criação da tabela "**usuario**", conforme este a seguir:

```
CREATE TABLE `usuario` (
  `id_usuario` int(4) NOT NULL auto_increment,
  `nome_usuario` varchar(50) NOT NULL,
  `e_mail_usuario` varchar(50) NOT NULL,
  `telefone_usuario` varchar(30) NOT NULL,
  PRIMARY KEY  (`id_usuario`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

Código SQL gerado pelo phpMyAdmin. Após criarmos a tabela **usuario**

Se o seu código SQL estiver igualzinho a este então voce está de parabéns nesta tarefa, podendo seguir em frente tranqüilamente. Se não está igual, faça as devidas correções, para evitar problemas posteriores.

Como já havia dito, nosso sistema terá inicialmente uma única tabela, para facilitar o aprendizado, mais à frente poderemos incluir outras tabelas e até falar um pouco sobre relacionamento entre tabelas. Aí sim voce entenderá um pouco mais sobre qual a utilidade da **chave primária** (id_usuario).

Fazendo a conexão com o banco de dados

Para conectar-se ao banco é necessário informar ao MySQL quem é o usuário que está se conectando, qual a senha e qual banco pretende conectar-se, levando em consideração que no MySQL pode-se ter vários bancos de dados. Na verdade o MySQL é um gerenciador de banco de dados.

O ideal é que se crie no MySQL um usuário para cada banco de dados que for criado, mas no nosso caso iremos utilizar o usuário **root**, que é o usuário principal, e não utilizaremos senha por enquanto. Sendo assim, nosso código PHP ficará da seguinte forma.

```
1  <?php
2      $hostname = "localhost";
3      $database = "dgestruturado";
4      $username = "root";
5      $password = "";
6      $con = mysql_connect($hostname, $username, $password)
7      or die(mysql_error()."Erro ao tentar conectar-se ao banco");
8
9      mysql_select_db($database, $con);
   ?>
```

- Explicando o código:

Na variável $hostname está o local onde meu MySQL e o banco de dados estão instalados. No nosso caso ele está na nossa própria máquina, então o local é "localhost". Se estivesse instalado em uma outra máquina, ao invés de localhost poderia ser colocado o IP (endereço) da outra máquina ou o nome.

Na variável $database está o nome do nosso banco "dgestruturado", que foi o nome que atribuímos ao banco quando o criamos no PhpMyAdmin.

Em $username está o nome do usuário que acessará o banco. Estamos utilizando aqui o usuário principal padrão do MySQL, depois veremos como criar um usuário para cada novo banco que criarmos.

A variável $password está com valor nulo (em branco), porque na instalação do Wamp ele atribui, por padrão, senha em branco, mas depois, quando estivermos desenvolvendo sistemas de verdade, poderemos atribuir uma senha para acesso ao banco, por questão de segurança.

A variável $con, faz a conexão com o banco utilizando o comando mysql_connect, do PHP. Observe o comando "or die", ele emitirá uma mensagem

de erro caso, por algum problema, não seja possível fazer a conexão com o banco. A mensagem de erro pode ser a que voce quiser colocar.

Na linha 9, temos um comando que não faz parte da conexão, mas que é responsável por selecionar o banco de dados que pretendemos acessar. Observe que neste comando temos a variável com o nome do nosso banco ($database) e a variável com a conexão ($con).

Estas variáveis também poderiam ter qualquer nome, mas estes são os mais utilizados.

TAREFA

Digite este código e salve em um arquivo com o nome de **conexao.php**, na pasta **www/dgestruturado**.

Abra o browser de internet e digite
http://localhost/dgestruturado/conexao.php

Se não aparecer nenhuma mensagem de erro é porque deu tudo certo.

Se não funcionar, tente descobrir onde está o erro, com o seu próprio esforço, então, se mesmo assim não conseguir, acesse o nosso forum e envie sua dúvida.

Com certeza aprendemos mais tentando descobrir os nossos erros
para corrigi-los, do que perguntando aos outros.

Exercite sua capacidade de ser um autodidata, mas mesmo assim não se intimide em nos enviar suas dúvidas. Elas podem serem úteis para melhorarmos ainda mais o texto desta apostila.

Próximo-passo

Agora abra o arquivo "**cadastro_usuario_exe.php**" que já havia sido criado antes, conforme está na página 8.

Neste arquivo inclua na primeira linha do código o seguinte comando, logo abaixo da instrução <?PHP

```
include ('conexao.php');
```

O **comando include** serve para incluir um arquivo dentro de outro. No nosso caso estamos incluindo o arquivo "**conexao.php**" dentro do arquivo "**cadastro_usuario_exe.php**".

Não se esqueça que o nome do arquivo, no comando include, tem que estar entre aspas e que todo comando em PHP termina com ponto-e-vírgula. Pode-se usar aspas simples ou duplas.

Agora faça novamente o teste abaixo:

Abra o browser de internet e digite
http://localhost/dgestruturado/cadastro_usuario.htm

Preencha o formulário e clique no botão "**Enviar**".

Se não der erro, prossiga, se acontecer algum erro tente descobri-lo. Tente interpretar as mensagens de erro do PHP, apesar das mensagens serem em inglês, ele geralmente indica o nome do arquivo e o número da linha que ocorreu o erro.

A utilização do comando "include" é uma forma de não ter que ficar repetindo um mesmo código php várias vezes. Por exemplo, quando inserimos o arquivo "conexao.php" dentro do arquivo "cadastro_usuario_exe.php" estamos na verdade fazendo a mesma coisa que copiar e colar o código do arquivo

conexao.php dentro do arquivo cadastro_usuario_exe.php. Ou seja, se voce simplesmente digitasse o código de conexão dentro do arquivo "cadastro_usuario_exe.php" também funcionaria, só que, se voce tiver que utilizar os comandos de conexão em outros arquivos, teria que digitá-los (ou copiar e colar) novamente e isso não seria uma boa prática de programação.

Para evitar a repetição de um mesmo código, utilize sempre o recurso de inclusão de arquivo. É importante também dividir por arquivos as tarefas específicas do sistema que voce estiver desenvolvendo. Por exemplo, arquivo de conexão deve ter apenas comandos de conexão com o banco; arquivo para gravar dados no banco deverá ter apenas comandos de gravação de dados, e assim por diante.

TAREFA:
- Para fixar ainda mais o seu aprendizado, faça uma lista dos comandos de PHP já aprendidos até aqui.
Faça isso como atitude de uma pessoa que quer ser disciplinada e esforçada no aprendizado de PHP. Tenho certeza que lhe será útil.

Iniciando os primeiros comandos SQL no PhpMyAdmin

Se voce já pesquisou na internet sobre MySQL conforme eu recomendei no início desta apostila, então esta parte da apostila deverá ser bem tranquila para aprender e praticar.

Voce já deve ter criado a tabela, conforme está sendo pedido na página 10 (início da segunda parte da apostila), vamos agora aprender a inserir dados nesta tabela.

Aprenderemos primeiro a utilizar os comandos SQL no PhpMyAdmin e depois veremos como utilizá-los no PHP.

Inserindo, Listando, Alterando e Apagando dados na tabela "usuario"

Lembre-se que os campos que foram criados para a tabela "usuario" deste nosso banco de dados são: id_usuario, nome_usuario, e_mail_usuario e telefone_usuario; Para se fazer a insersão de dados em uma tabela do banco é necessáiro saber em quais campos se deseja inserir os dados.

COMANDO PARA INSERSÃO DE DADOS NA TABELA "USUARIO"

```
INSERT INTO usuario (nome_usuario, telefone_usuario)
VALUES ('Maria da Graça', '3334-5678')
```

Explicando o comando:

- Observe que os comando SQL coloquei em letras maiúscula, isto não é necessário, é apenas por questão de melhor visualizar os comandos SQL. Recomendo que façam sempre assim, por simples questão de organização e melhor visualização do seu código. Lembre-se que, quando melhor estiver

organizado o seu código, mais fácil será identificar erros e problemas no seu programa.

- Logo após o "INSERT INTO" vem o nome da tabela que receberá os dados. No nosso caso é a tabela "usuario". E logo após o nome da tabela vem os nomes dos campos que receberão dados. Se eu fosse inserir apenas o nome da pessoa e não quisesse inserir o telefone, eu **não colocaria** alí o nome do campo "telefone_usuario". Observe que não inclui o campo "e_mail_usuario", portanto isso quer dizer que não vou cadastrar, por enquanto, o e-mail do usuário.
- Após o nome dos campos vem o comando VALUES, para informarmos os valores que serão inseridos nos campos. Observe que são dois campos que receberão valores e que estes valores são letras (strings), por se tratar de letras é necessário que estes valores estejam **entre aspas**.
- Observe também que **não foi necessário** colocar o campo "id_usuario", este é um **campo de identificação única** da tabela "usuario" ele é um campo que recebe numeração automaticamente. Reveja o script de criação da tabela que está no início da página 12 e observe que na linha do "id_usuario" há um parâmetro chamado "`auto_increment`", este parâmetro indica que este campo receberá automaticamente um valor numérico e sequencial. Este campo é chamado de **CHAVE PRIMÁRIA**.

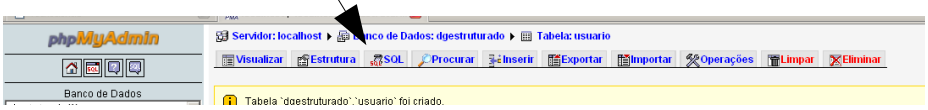
DETALHE IMPORTANTE:

A Chave primária neste caso é um campo com numeração automática, que tem uma particularidade: voce verá que ao inserir o primeiro registro na tabela "usuario", este campo (id_usuario) receberá o valor "1", e ao inserir mais um registro, este campo receberá automaticamente o valor "2", no entanto se logo depois você excluir o registro 2 do banco de dados, e em seguida inserir um novo registro, este novo registro receberá o número "3", mesmo não existindo mais o registro número 2. Isso acontece por questão de segurança do próprio banco de dados, de modo a evitar, de todas as formas, a existência de dois cadastros com um mesmo número de CHAVE PRIMÁRIA.

TAREFA:

Agora que já entendeu o comando INSERT execute-o no PhpMyAdmin, conforme orientação abaixo:

- Acesse o PhPMyAdmin conforme orientação da página 5 (*clicando no ícone do Wamp e acessando PhpMyAdmin*).
- No PhpMyAdmin, selecione o banco que criamos com o nome de **dgEstruturado** (*basta clicar no nome do banco que aparece à direita*).
- Clique na aba "SQL".



- Digite o comando "INSERT INTO" conforme foi mostrado na página 14.
- Clique no botão "Executar", que aparece no canto direito da tela.
- Observe se o comando foi executado com sucesso, ou se ocorreu alguma mensagem de erro. Mesmo se for uma mensagem em inglês, veja que não é difícil interpretá-la. Se quiser mesmo aprender programação, vá se acostumando com o inglês, pois ele é muito utilizado na informática em geral. A maior parte do conteúdo sobre programação na internet está em inglês.

LISTANDO

Agora vamos verificar se os dados foram mesmo inseridos no banco.

- Basta clicar na aba "Visualizar" (é a primeira aba). Verifique então que logo abaixo será **listado** o registro que voce acabou de inserir no banco, conforme imagem abaixo:

	id_usuario	nome_usuario	e_mail_usuario	telefone_usuario
	1	Maria da Graça		3334-5678

ALTERANDO E EXCLUINDO REGISTROS

Observe os seguintes botões que aparecem na figura anterior:

O lápis é o link para acessar o formulário em que é possível fazer alterações nos dados do registro cadastrado. Por exemplo, **alterar** o nome da pessoa ou incluir o endereço de e-mail.

O "X" é para **excluir** um registro.

Observem também que ao clicar na aba "Visualizar" o phpMyAdmin, para fins didáticos, ele mostra como ficaria o comando em SQL para listar na tela o conteúdo da tabela. O mesmo também se voce fizer alteração ou exclusão de um registro.

Agora que já conhecemos os comandos para inserir, alterar, apagar e listar registros no PhpMyAdmin, veremos mais adiante, como utilizá-los no código PHP.

Iniciando os primeiros comandos SQL no PHP

Aqui começa a ficar um pouco mais complicado, mas ainda não é tão difícil assim. Não é nada que você não consiga fazer. Tendo calma, muita atenção e paciência consegue-se tudo aqui.

Seu arquivo "**cadastro_usuario_exe.php**" já deverá estar então com o comando "**include 'conexao.php';**", que faz a inclusão do arquivo de conexão. A partir daí iremos inserir as seguintes linhas de comando **no final** do arquivo "**cadastro_usuario_exe.php**":

```
$sql="INSERT INTO usuario (nome_usuario, e_mail_usuario, telefone_usuario)
            VALUES ('".$nome_usuario."','".
                        $e_mail_usuario."','".$telefone_usuario."')";
$result = mysql_query($sql);

if ($result)
   echo "Dados cadastrados com sucesso!";
   else
      echo "Erro ao tentar cadastrar dados no banco!";
```

Explicando o código:

Estamos utilizando aí, duas variáveis: $sql e $result. Lembre-se que, nomes de variáveis sempre iniciam com o sinal "$". **Pode-se atribuir qualquer nome às variáveis**, mas, por questão de padronização e por uma melhor organização,

sempre utilizamos os nomes $sql e $result para utilização de comandos de banco de dados.

Observe que a variável **$sql** recebeu como valor um comando SQL de inserção de dados, que, se observar bem, vai ver que é o mesmo comando que utilizamos lá no **PhpMyAdmin**, conforme está na página 14.

Compare este comando SQL com os comandos SQLs que aparecem lá no PhpMyAdmin e verá que são praticamente iguais.

Observe também que as variáveis que aparecem no comando INSERT INTO, logo depois da palavra "VALUES", estão entre aspas simples. Observe que coloquei as aspas simples em vermelho para ajudá-lo a percebê-las e não esquecer de inclui-las. No comando SQL, os valores a serem inseridos no banco tem que estar entre aspas simples. Observe que utilizei concatenação (*veja na página 9*)

A variável $result pega o conteúdo da variável $sql e envia para o banco de dados utilizando o comando mysql_query().

Existem várias outras formas de se fazer inserção de dados no banco utilizando PHP, esta é apenas uma delas e acredito que é a mais simples.

Se não souber para que servem os **comandos IF e ELSE**, pesquisa na internet, é muito fácil entendê-los, eles estão presentes em todas as linguagens de programação.

TAREFA:

Agora que já entendeu este comando e já incluiu as linhas acima, no seu arquivo, conforme recomendei. Faça o teste: acesse o seu formulário, preencha os dados e clique no botão "Enviar". Depois acesse o phpMyAdmin e verifique se os dados foram realmente gravados.

Se aparecer mensagens de erro, verifique na mensagem, qual a linha do seu programa está dando erro e faça a correção.

Como já disse anteriormente, tente sempre interpretar as mensagens de erro do PHP, elas são muito úteis para ajudá-lo a descobrir o erro. Mas não se preocupe em traduzir o texto em inglês para saber exatamente o que ela quer dizer, apenas tente capturar na mensagem as informações mais importantes.

Para fazer o cadastro também é necessário que criemos um código que faz uma verificação para ver se os dados que estão sendo cadastrados estão corretos. Por exemplo, para cadastrar uma pessoa, o campo do nome da pessoa não pode ficar em branco, e também não se pode aceitar digitar qualquer coisa no campo de e-mail ou do telefone. Mas isso nós não iremos ver agora, para evitar maiores complicações. O objetivo aqui é mostrar, de forma simples, como as coisas funcionam. Bem mais à frente, quando estivermos criando sistemas mais sofisticados, poderemos aprender essa parte.

LISTANDO OS DADOS QUE FORAM INSERIDOS NO BANCO

Primeiramente vamos ver como fazer para listar todos os registros cadastrados no banco, depois veremos como listar apenas um registro específico ou como criar o formulário para alteração dos dados.

Como eu já disse anteriormente, é recomendável que tenhamos uma página para cada função do nosso sistema. Então, se já temos a página que inclui os dados no banco de dados, vamos criar agora a página que vai listar estes dados em ordem alfabética.

TAREFA:

Crie um arquivo novo chamado **listar_usuarios.php.**

Obs.: Não utilize acentuação nos nomes dos arquivos, isso pode causar alguns problemas.

Neste novo arquivo inclua o seguinte código de programação:

```
1  <?php
2  include ('conexao.php');
3
4  $sql = "SELECT * FROM usuario";
5  $result = mysql_query($sql);
6  $row = mysql_fetch_array($result);
7  ?>
8  <html>
9    <body>
10     <table border='1'>
11       <tr>
12         <td>Nome</td>
13         <td>E-mail</td>
14         <td>Telefone</td>
15       </tr>
16       <?php
17         do {
18           echo "<tr>";
19            echo "<td>".$row['nome_usuario']."</td>";
20            echo "<td>".$row['e_mail_usuario']."</td>";
21            echo "<td>".$row['telefone_usuario']."</td>";
22           echo "</tr>";
23         }while ($row = mysql_fetch_array($result));
24       ?>
25     </table>
26     <a href="index.php">voltar</a>
27   </body>
28 </html>
```

Explicando o código:

Temos bastante coisa para falar aqui.

Na linha dois temos o include que colocamos no arquivo anterior, lembra? Assim estamos evitando ter que digitar novamente o código de conexão.

Observe que deixei uma linha em branco depois do comando include, isso ajuda muito a deixar o código do seu programa mais fácil para ler. Deixe linhas em branco para separar comandos diferentes.

Na linha quatro tem o comando SELECT, que, na verdade é o mesmo que utilizamos no PhpMyAdmin.

A linha cinco, como já disse anteriormente, é a função que vai enviar o comando SQL para o banco de dados.

Aqui começa a novidade. Como se trata de um comando SELECT, a variável $result (linha 5), ao enviar o comando para o banco, receberá de volta o conteúdo da tabela usuario. Na linha seis temos a variável $row, que receberá o conteúdo da variável $result e **transformará** este conteúdo em um array.

Mas o que é um array?

Recomendo-lhe pesquisar no google sobre este assunto (digite no google "array + PHP"), mas de início lhe adianto que array é uma estrutura de dados. Da mesma forma que podemos atribuir um valor a uma variável, podemos também atribuir mais que um valor a uma mesma variável se utilizarmos um array. No exemplo acima a variável $row recebeu todos os campos da tabela "usuario", ou seja, a variável $row recebeu mais de uma valor. Para conseguirmos ver estes valores basta observar os comandos que se seguem nas linhas 19 a 21.

Quer ver um exemplo de array?

Então observe o código a seguir:

```
<?php
 $a['nome']="Maria Aparecida";
 $a['idade']=32;

 echo "NOME: ". $a['nome']."<br>";
 echo "IDADE: ". $a['idade']."<br>";
?>
```

Exempo do uso de array

Observe que a variável $a é a mesma para os dois itens (nome e idade), só que ela é um array com dois elementos 'nome' e 'idade'.

Observe que, por se tratar de array utilizamos "[ ]" (colchetes) e não parênteses. Os nomes que estão dentro dos colchetes estão também entre aspas simples.

Uma outra novidade no código acima é o uso do comando do... while. Esse comando, traduzindo para o português, quer dizer "faça isso, enquanto..." ou, "enquanto tiver algo para fazer, faça".

Da linha 18 a 22, estamos dizendo para o PHP imprimir todo o conteúdo da tabela "usuario". Ele faz isso da seguinte forma: da linha 19 a 21 ele vai imprimir o primeiro registro cadastrado na tabela "usuario". Ao chegar na linha 22 (while...) ele vai verificar se existe mais algum registro na tabela usuario, se houver, este comando irá pular para o próximo registro da tabela "usuario" e voltará para a linha de comando 18, até que ele consiga imprimir todo o conteúdo da tabela.

Entendeu bem? Se não, envie-me sua dúvida. Ou deixe que, com a prática voce vai aprendendo aos poucos. As vezes não é tão importante que se entenda tudo de uma só vez. À medida que voce for praticando, vai assimilando os comandos aos poucos e entendendo o seu funcionamento.

TAREFA:

Vamos voltar agora para o HTML. Iremos criar mais uma página que nos possibilitará testar melhor nossos comandos PHP.

Crie um arquivo com o nome **index.php**

Neste arquivo inclua o seguinte código:

```
<html>
   <body>
      <p>Página Inicial</p>
      <p><a href="cadastro_usuario.htm">Cadastrar</a></BR>
      <p><a href="listar_usuarios.php">Listar</a></p>
    </body>
</html>
```

Pronto! - enfeite a página como quiser. O mais importante aí são os links que criamos (<a href>).

Agora para acessar o seu sistema, basta digitar no seu browser o seguinte:

http://localhost/dgestruturado

O PHP irá abrir a página **index.php**.

Todo programa em PHP que voce desenvolver, ou todo site que voce criar, sempre terá que ter um arquivo chamado index.php ou index.html. Este arquivo é sempre o primeiro arquivo a ser aberto por um site ou sistema de internet.

Ao abrir esta página no seu browser, verá que aparecerá uma tela com os dois links que criamos: um para acessar o formulário de cadastro e outro para acessar a lista de usuários cadastrados.

Clique nos links e veja o que acontece.

Se surgirem mensagens de erro, observe qual a linha que está dando erro e faça a correção. Isso é normal para um programador iniciante.

Depois veremos como melhorar a navegação no nosso sistema. Incluiremos links para voltar à página inicial. Se voce já souber fazer isso, então já faça.

Se ainda não é muito bom em html, pratique html o quanto puder. Quanto mais souber html mais interessante ficarão as páginas do seu sistema.

O html é para a parte visual do seu sistema e o PHP para as funcionalidades e acesso ao banco de dados. Lembra?

Entendeu tudo até aqui?

- Se entendeu, então continue adiante. Se não entendeu, releia tudo novamente e pratique tudo o que está pedindo, porque deste ponto em diante vai só ficando mais complexo e aumentando o grau de detalhes.

ALTERANDO E EXLCUINDO DADOS

OK. Voce então já aprendeu a Incluir dados e listá-los. Agora vamos aprender a Alterar estes dados e Exclui-los.

Deve se ter muito cuidado com comandos de alteração e exclusão, pois são comandos que podem causar danos irreversíveis ao seu banco de dados, se forem utilizados de maneira incorreta. Um erro de programação pode levar tudo a perder.

Os comandos de alteração, por exemplo, são normalmente utilizados para fazermos alterações em apenas um único usuário cadastrado no nosso banco, mas por algum descuido no comando, podemos causar alteração em todos os registros de uma só vez.

No PHPMyAdmin, para fazer alteração em um registro, basta listar os registros e clicar no ícone do lápis que aparece ao lado dos registros, - lembra?

E para excluir, clica-se no "X" vermelho.

Vamos ver agora como fazer isso no PHP.

TAREFA:

Abra o arquivo "cadastra_usuario.htm" e salve-o com outro nome (clique em salvar como... e digite **altera_usuario.php**).

Pronto! O arquivo que voce tem aberto na sua tela agora é o altera_usuario.php. Certo?

Neste arquivo faremos algumas modificações. Na página seguinte está o código, com as modificações a serem feitas. Coloquei as modificações em

vermelho para ficar mais fácil visualizar o que precisa ser acrescentado ao arquivo altera_usuario.php

```
1  <?php
2    include ('conexao.php');
3
4    $id_usuario=$_GET['id_usuario'];
5
6    $sql="SELECT * FROM usuario WHERE id_usuario='$id_usuario' ";
7    $result = mysql_query($sql);
8    $row = mysql_fetch_array($result);
9  ?>
10 <html>
11    <body>
12       <p>Cadastro de Uusários</p>
13       <form name="cadastro de usuario" method="post" action="altera_usuario_exe.php">
14          <table>
15             <tr>
16               <td>Nome:</td>
17               <td><input name="nome_usuario" type="text" size="50" maxlenght="50" value="<?php echo
18 $row['nome_usuario']?>">
19               </td>
20             </tr>
21             <tr>
22               <td>E-mail:</td>
23               <td><input name="e_mail_usuario" type="text" size="50" maxlenght="50"
24                     value="<?php echo $row['e_mail_usuario']?>"></td>
25             </tr>
26             <tr>
27               <td>Telefone:</td>
28               <td><input name="telefone_usuario" type="text" size="30" maxlenght="30"
29                     value="<?php echo $row['telefone_usuario']?>"></td>
30             </tr>
31             <tr>
32               <input name="id_usuario" type="hidden" value="<?php echo $row['id_usuario']?> ">
33               <td colspan=2 align="center"><input type="submit" name="Submit" value="Alterar"></td>
34             </tr>
35          </table>
36       </form>
37    </body>
   </html>
```

Explicando o código:

Observe as seguintes modificações que foram feitas:

- Da linha 1 até a linha 9, temos apenas código PHP.
- Na linha 13, o "action" agora está chamando o arquivo **altera_usuario_exe.php**, teremos que criar este arquivo depois.
- Em cada um dos campos do formulário, incluimos o atributo "value" e, a este atributo incluimos um código PHP.
- Na linha 31 criei um outro campo. É o campo "id_usuario" este campo está oculto, observe que o valor do atributo "type" é "hidden" (oculto). Este campo servirá para indicar ao MySQL que apenas um registro deverá ser alterado.
- O botão submit, que fica no final do formulário, o valor passou a ser "Alterar".

Veja que o select que criamos na linha 6 é muito parecido com o select que temos no arquivo **listar_usuarios.php**. A diferença é um WHERE que acrescentamos agora. Este WHERE é para informarmos ao MySQL que deverá ser listado apenas um registro.

Explicarei com mais detalhes após fazermos a próxima tarefa. Teremos que fazer uma modificação no arquivo listar_usuarios.php.

TAREFA:

– Abra o arquivo **listar_usuarios.php**,

– Na linha 20 (ou 19), onde está o seguinte: `echo "<td>".$row['nome_usuario']."</td>";`

Nesta linha iremos criar um link para, quando clicarmos no nome do usuário, o sistema deverá abrir a página de alteração de dados (altera_usuario.php).
Para fazer isso, modifique essa linha para:

```
echo "<td><a href='altera_usuario.php?id_usuario=".$row['id_usuario']."'>".$row['nome_usuario']."</a></td>";
```

Aqui tem aspas simples junto com aspas duplas.

Essa linha ficou meio difícil de digitar por que tem algumas aspas que se confundem. Tem aspas simples junto com aspas duplas. Por esse motivo, ao invés de digitar toda essa linha, recomendo copiá-la e colá-la no arquivo listar_usuario.php, no lugar certo.

Ao copiar e colar, observe se vai acontecer alguma alteração nas aspas. Geralmente elas ficam diferentes, e isso causa problema no código. Se acontecer, altere apenas as aspas. Elas estarão inclinadas para a esquerda, substitua por aspas normais.

CRIANDO O ARQUIVO altera_usuario_exe.php

O início deste arquivo vai ser idêntico ao início do arquivo cadastra_usuario_exe.php. Apenas inclui uma linha à mais que está destacada em vermelho.

O conteúdo dele será o seguinte:

```
<?php
  include ('conexao.php');

 $id_usuario=$_POST['id_usuario'];
 $nome_usuario=$_POST['nome_usuario'];
 $e_mail_usuario=$_POST['e_mail_usuario'];
 $telefone_usuario=$_POST['telefone_usuario'];

 echo "<P>Nome do usuário: ".$nome_usuario."<BR>";
 echo "E-mail: ".$e_mail_usuario."<BR>";
 echo "Telefone: ".$telefone_usuario."</P>";

          $sql="UPDATE usuario  SET
                    nome_usuario=."'$nome_usuario."',
                    e_mail_usuario=."'$e_mail_usuario."',
                    telefone_usuario=."'$telefone_usuario."'
                WHERE id_usuario=."'.$id_usuario."';
          $result = mysql_query($sql);

          if ($result)
            echo "Dados alterados com sucesso!";
            else
              echo "Erro ao tentar alterar dados no banco!";
?>
<a href="index.php">voltar</a>
```

Comando do MySql para alteração

Este WHERE é para informar que apenas um registro será alterado.

Observe o uso de aspas simples junto com aspas duplas.

Esta última linha é um link para voltar para a página inicial do sistema.

- Observe também a linha que destaquei em vermelho (logo abaixo do comando include), a terceira linha. Ela está pegando do formulário o valor do campo "id_usuario". Lembre-se que este é um campo oculto no formulário. Pois bem, esta variável é colocada no comando WHERE do SQL, conforme está indicando o segundo quadro azul acima. A variável "id_usuario" guarda o valor de identificação do registro no banco de dados (é a chave primária na tabela do

banco de dados). Esta variável serve para informar ao comando SQL, qual é o registro que deverá ser alterado. E somente este registro será alterado.

Até aqui voce já deve ter aprendido bastante coisa. Falta agora apenas aprender a excluir registros, que é um processo mais simples que os anteriores.

EXCLUINDO DADOS

Para fazer uma exclusão de dados, é necessário que tenhamos uma tela pedindo ao usuário que confirme se realmente quer excluir o registro, pois, ao excluir um registro, não tem mais como recuperá-lo.

Como estamos apenas aprendendo os primeiros passos, não iremos fazer isso agora. Nosso sistema irá excluir sem perguntar nada. Mais adiante, quando estivermos desenvolvendo sistemas mais modernos, colocaremos este recurso.

Para fazer a exclusão iremos acrescentar mais duas linhas ao arquivo "listar_usuarios.php".

TAREFA:

Abra o arquivo **listar_usuarios.php**.

Depois de linha 14 (<td>Telefone</td>) inclua a linha: <td>Excluir</td> da seguinte forma:

Depois da linha (echo "<td>".$row['telefone_usuario']."</td>";) crie a seguinte linha:

```
echo "<td><a href='excluir_usuario.php?id_usuario='".$row['id_usuario']."'>Excluir</td>";
```

Aspas duplas junto de aspas simples

Agora crie o arquivo "excluir_usuario.php", com o seguinte código:

```
<?php
  include ('conexao.php');

      //id_usuario vem do arquivo listar_usuario.php
      $id_usuario=$_GET['id_usuario'];

     $sql="DELETE FROM usuario WHERE id_usuario='".$id_usuario."'";
     $result = mysql_query($sql);

      if ($result)
         echo "Registro excluído com sucesso!";
         else
            echo "Erro ao tentar excluir registro no banco!";
?>
<a href="index.php">voltar</a>
```

Protinho. Até aqui vc já deverá ter aprendido o básico de incluir, alterar, listar e excluir registros num banco de dados MySql utilizando PHP.

Se conseguiu, meus parábens. Se ainda não entendeu muito bem, ou se estiver com alguma dificuldade, pode contar comigo, ou releia a apostila e tente descobrir o que está faltando no seu código.

Nos próximos textos começaremos a falar sobre Programação Orientada a Objetos, numa visão bem básica do que vem a ser isso. E depois, poderemos ir aprimorando aos poucos nossos conhecimentos.

Na internet tem um forum muito bom para estudo e para enviar suas dúvidas.
Acesse: http://www.phpbrasil.com/phorum/index.php

PROGRAMANDO UTILIZANDO ORIENTAÇÃO A OBJETO (O-O)

4a. parte

- Em programação, o que entendemos que é um Objeto?

A grosso modo, eu diria que objeto é por exemplo, uma caixa de ferramentas. E dentro desta caixa temos ferramentas específicas para cada tipo de ação que precisaríamos fazer. Na vida real por exemplo, poderemos ter várias caixas de ferramentas, uma para cada tipo de trabalho, por exemplo: teríamos uma caixa para trabalho de marcenaria, uma outra caixa de ferramentas para manutenção elétrica, ou outra para mecânica de automóvel, etc. Cada uma delas tem ferramentas diferenciadas e as vezes até bem parecidas.

Na programação podemos ter então, vários objetos, cada um deles tendo ferramentas específicas para cada tipo de atividade. Iremos criar arquivos/objetos específicos para cada tipo de ação ou elementos do nosso sistema.

- Entendeu essa parte? Se não releia, é importante que entenda.

Torno a lembrar aqui, que a intensão desta apostila é ser o mais prático possível, deixando um pouco a teoria de lado, mas aconselhando sempre a pesquisar mais sobre o assunto na internet.

Orientação a objeto é um assunto bem mais complexo do que o conteúdo que estou apresentando neste início de conversa, mas, com certeza, se voce entender bem estes conceitos iniciais, terá mais facilidade para entender boa parte de O-O.

Voce perceberá que, no início da criação dos seus códigos, vai se levar muito mais tempo para programar utilizando O-O do que desenvolver programas de forma estruturada, como o que fizemos anteriormente, no entanto, a manutenção em um sistema O-O fica muito mais fácil e tranquila, ainda mais se for um sistema complexo.

No nosso caso, estamos desenvolvendo um pequeno sistema que grava, altera, lista e exclui dados. Em um sistema simples como esse, não faz muito sentido utilizarmos orientação a objeto, até porque a verdadeira vantagem da O-O é a facilidade que se tem em fazer manutenção (alterações na programação) em sistemas muito grandes e complexos. Mas, só para fins didáticos, iremos refazer o nosso sistema utilizando O-O. Depois, poderemos desenvolver novas funcionalidades, fazendo com que ele fique um pouco mais complexo e mais útil.

Se o nosso sistema trabalha com os dados dos usuários, gravando-os na tabela "usuarios" do nosso banco de dados, então teremos que ter um "**objeto**" que cuidará de tudo o que se referir a usuário. Este objeto se chamará "Usuario", assim fica fácil saber que é um objeto que cuida dos dados dos usuários.

Dentro deste objeto teremos as **funções** (ferramentas) específicas para cada ação realizada pelo nosso sistema (entenderemos funções aqui como sendo as ferramentas dentro da caixa de ferramentas). Sendo assim, dentro do objeto "usuario" deveremos ter uma função para gravar, outra para alterar, outra para listar e outra para excluir, conforme mostrado na seguinte figura:

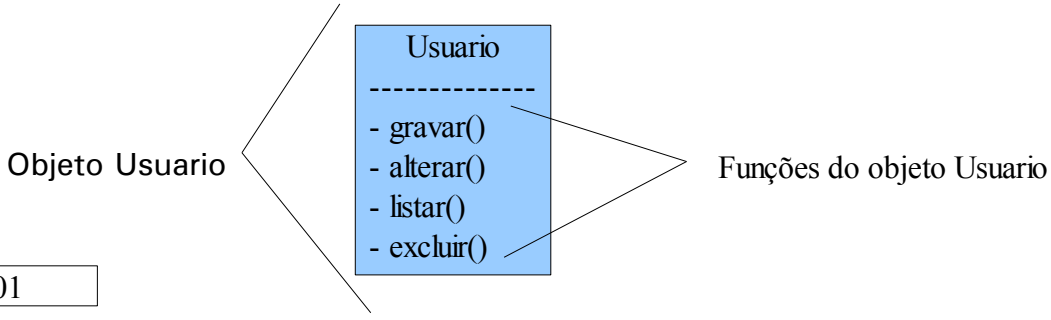


Fig. 01

TAREFA:

1. Na pasta "www" crie uma pasta chamada "**dgoo**" (abreviação de DataGrid Orientado a Objeto), um nome que inventei.
   Dentro desta pasta iremos criar nosso sistema O-O.
2. Copie para dentro dela os arquivos dos formulários que criamos no sistema anterior. Copie também o arquivo que faz a conexão com o banco de dados e o arquivo da página inicial. Os arquivos são: **conexao.php**, **cadastro_usuario.html**, **altera_usuario.php** e **index.php**.

COMO CRIAR UM OBJETO?

O mais importante em programação O-O é saber programar utilizando Classes e Funções.

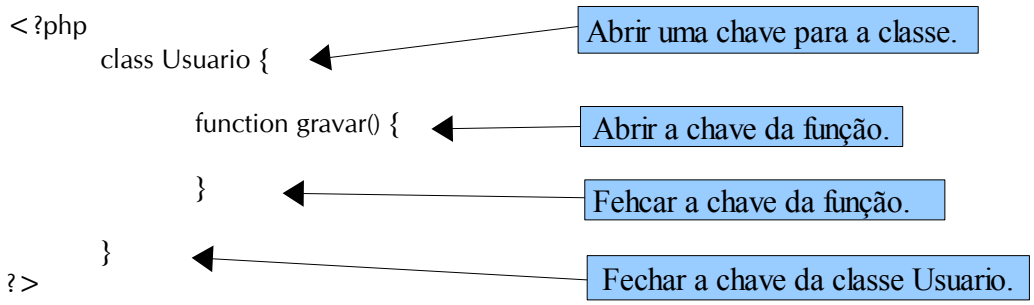
O que é uma classe? - Inicialmente, vamos entender da seguinte forma: "uma classe é um objeto". Chamaremos de "classe" o objeto que criaremos na nossa programação. Acontece que, um objeto pode conter várias classes, mas isso não vem ao caso agora. Depois explicarei melhor, por enquanto fica entendido que, **um objeto é uma classe**. Certo?

Voce ainda recorda o que é um objeto? Lembra da analogia que fizemos com a caixa de ferramentas? Pois é, mantenha esse conceito na sua cabeça, isso é importante.

Pois bem, a classe é a "caixa de ferramentas" e dentro dela teremos as nossas ferramentas, que, em programação, chamaremos estas ferramentas de **funções,** ou **métodos**. Na figura mostrada anteriormente (Fig. 01), temos o exemplo de uma classe e suas funções, trata-se da classe "Usuario", a qual possui as funções (ou métodos) para gravar, alterar, listar e excluir.

Vamos agora ver como é esta classe com seus respectivos métodos.

Para criar uma classe utiliza-se o comando "**class**" e para criar as funções utiliza-se o comando "**function**", conforme está sendo mostrado abaixo:



Observe que, quando criamos uma classe devemos abrir uma chave e fechá-la logo abaixo. Dentro destas chaves deveremos ter as funções. Observe que ao criarmos uma função também devemos abrir uma chave e fechá-la. Neste caso a função gravar está dentro da classe Usuario.

Observe também que o nome da classe deve iniciar-se com letra maiúscula. Isso não é uma obrigatoriedade, mas é um consenso entre os programadores, portanto vamos tentar segui-lo, para manter nosso código o mais próximo possível dos padrões utilizados em programação O-O. É importante também fazer estes espaçamentos que fiz, assim fica mais fácil a visualização de que uma coisa está dentro da outra.

Outro detalhe que quase nos passa despercebido são os parênteses que colocamos nas funções. Você percebeu isso? Pois bem, estes parênteses são a "porta de entrada" (vamos chamar assim) para as suas funções. **Um função deve "quase sempre" receber informações, processar e entregar um resultado.** Veremos isso mais adiante.

Vamos criar então, nossa primeira classe.

TAREFA:

Digite o seguinte código:

```
<?php
        class Usuario_dao {

                function gravaUsuario( ) {

                }

                function alteraUsuario( ) {

                }

                function listaUsuario( ) {

                }

                function excluiUsuario( ) {

                }

        }
?>
```

"Porta de entrada" da função.

Salve este arquivo com o nome **usuario_dao.php**. Depois lhe explico melhor porque colocamos "**_dao**" no nome deste arquivo.

Pronto! - Aqui temos nossa classe "Usuario_dao" com todas as suas funções, falta agora colocarmos os comandos dentro das nossas funções.

Para colocar estes comandos, temos que, basicamente, copiar os comandos do programa estruturado que fizemos anteriormente e colar dentro da função. Fazendo isso, teremos a função "gravaUsuario" da seguinte forma:

```
function gravaUsuario( $nome_usuario,$e_mail_usuario,$telefone_usuario ) {

  include ('conexao.php');

   $sql="INSERT INTO usuarios (nome_usuario, e_mail_usuario, telefone_usuario)
      VALUES ('".$nome_usuario."','".$e_mail_usuario."','".$telefone_usuario."')";
  $result = mysql_query($sql);

  if ($result)
        return "Dados cadastrados com sucesso!";
     else
           return "Erro ao tentar cadastrar dados no banco!";
}
```

Este é o mesmo comando utilizado no programa estruturado que fizemos anteriormente.

Explicando o código:

Observe a "porta de entrada" da função. Nela temos as três variáveis que estão sendo utilizadas no comando INSERT INTO.

Observe também que há um comando novo neste código acima. O comando **return**. É ele que retornará o resultado depois que processar o comando de gravação. Logo mais veremos isso com mais detalhes e aí voce entenderá melhor. Observe as mensagens que o comando "return" envia.

Por uma questão de padronização os programadores também estabeleceram que a classe deve ter o mesmo nome do seu arquivo. Neste caso a classe **Usuario_dao** está gravada no arquivo **Usuario_dao.php.**

COMO UTILIZAR UM OBJETO?

TAREFA:

Só para fazermos um teste, copie do programa anterior o arquivo **cadastro_usuario_exe.php** e cole dentro desta sua nova pasta ("**dgoo**").

Abra este arquivo e faça as seguintes alterações nele:

- Apague neste arquivo, todas as linhas que são iguais às que estão dentro da função gravaUsuario(). Neste caso, são estas aqui:

```
$sql="INSERT INTO usuarios (nome_usuario, e_mail_usuario, telefone_usuario)
            VALUES ('".$nome_usuario."','".$e_mail_usuario."','".$telefone_usuario."')";
$result = mysql_query($sql);

if ($result)
   echo "Dados cadastrados com sucesso!";
  else
      echo "Erro ao tentar cadastrar dados no banco!";
```

- E, neste mesmo arquivo, no lugar das linhas que foram apagadas, digite o seguinte:

```
include ('Usuario_dao.php');
$grava = new Usuario_dao();
echo $grava->gravaUsuario($nome_usuario,$e_mail_usuario,$telefone_usuario);
```

- Apague também neste arquivo a linha include ('conexao.php'); Esta linha já está dentro da função gravaUsuario().

(Esteja certo que voce fará isso dentro do arquivo cadastro_usuario_exe.php, que voce acabou de copiar e que voce está dentro da pasta "dgoo").

Explicando o código:

O comando include voce já conhecer certo? Então vamos para os outros.

A variável **$grava**, está recebendo como valor, o conteúdo da classe Usuario_dao. Para que isso aconteça foi utilizado o comando **new**. Chamamos isso de "instanciar uma classe". Quando utilizamos o comando "new" estamos criando uma instância da classe. Este conteúdo fica temporariamente armazenado dentro da variável $grava, para que possa ser utilizado posteriormente. Isso é

instanciar. Podemos entender que é como pegar a caixa de ferramentas para ser utilizada.

Entendeu bem isso? É muito importante que entenda, isso é O-O.

Na linha seguinte (echo $grava...), observe que estamos utilizando o conteúdo da variável $grava para chamar a função gravaUsuario. Esta função, ao ser executada, retorna uma mensagem informando se os dados foram gravados ou não. Lembra do comando return que colocamos dentro da função?

- Observe que na função gravaUsuario() temos o comando return. É ele quem retorna a mensagem. Para fazer a mensagem aparecer na tela, foi utilizado na terceira linha, o comando "echo".

Agora faça o teste para ver se este seu novo sistema está gravando os dados no banco de dados. Se gravar aparecerá na tela a mensagem "dados Cadastrados com sucesso!".

Qualquer ocorrência de erro, revise o seu código de programação ou envie-me sua dúvida.

Parte 5

Quero lhe chamar-lhe a atenção para o seguinte problema:

– Na função "gravar" estamos passando três variáveis pela sua porta de entrada. Correto? Me diga agora, o que teríamos que fazer se no nosso formulário tivéssemos uns 50 campos para serem preenchidos, e que na nossa tabela do banco de dados tivéssemos uns 50 campos também. Como faríamos para gravar estes 50 campos no banco de dados?

– Voce então me responderia: - fácil – é só colocar as 50 variáveis na porta de entrada da função "gravar". Correto?

– Pois é, pode até ser, mas seu código de programação iria ficar muito feio e complicado com esse tanto de variável alí, além de que não é o que faria um bom programador, ou um programador profissional.

– A boa notícia é que a Orientação a Objeto tem uma solução para isso.

A classe Bean

A solução mais utilizada para este problema é a criação da classe "bean". Esta classe serve para juntar em um único objeto, todas as variáveis do seu sistema. Vamos dizer que seja uma caixa de ferramentas para guardar as variáveis com seus respectivos valores recebidos do formulário. Ela funciona quase como um array (lembra quando falamos sobre array?), só que uma classe é um pouco mais sofisticada que um simples array.

Se estamos trabalhando com os dados do usuário, nossa classe "bean" então se chamará "**Usuario_bean**". Dentro desta classe deveremos ter as variáveis do nosso sistema e as funções que vão receber os dados das variáveis. Esta classe deve ter uma variável para cada campo que temos na tabela "usuario".

TAREFA:

Digite e salve num arquivo com o nome "**Usuario_bean.php**", o seguinte código:

```
<?php
        class Usuario_bean {

            protected $id_usuario;
            protected $nome_usuario;
            protected $e_mail_usuario;
            protected $telefone_usuario;

            //GETs
            function getId_usuario() {
                return $this->id_usuario;
            }

            function getNome_usuario() {
                return $this->nome_usuario;
            }

            function getEmail_usuario() {
                return $this->e_mail_usuario;
            }

            function getTelefone_usuario() {
                return $this->telefone_usuario;
            }

             //SETs
            function setId_usuario($bean) {
                $this->id_usuario = $bean;
            }

            function setNome_usuario($bean) {
                $this->nome_usuario = $bean;
            }

            function setEmail_usuario($bean) {
                $this->e_mail_usuario = $bean;
            }

            function setTelefone_usuario($bean) {
                $this->telefone_usuario = $bean;
            }

        } //fechamento da classe
?>
```

- Explicando o código:

Observe que antes de criarmos as funções, criamos as variáveis. O termo "`protected`", que está antes dos nomes das variáveis, será explicado posteriormente. Vamos falar mais adiante, para não complicar muito agora.

Nesta classe temos dois tipos de funções: funções GETs e funções SETs.

Lembre-se que utilizo duas barras para escrever comentários no código. Como já devo ter dito antes, estas barras não interferem no funcionamento do código. Observe que nesta classe fiz três comentários. Os dois primeiros servem para informar quais são as funções GETs e quais são as SETs.

As funções GETs servem para passar para o programa os valores das variáveis e as funções SETs servem para pegar os valores das variáveis. Logo mais adiante, quando eu mostrar um exemplo da utilização desta classe "bean" voce entenderá melhor a diferença entre estes dois tipos de funções.

O comando "$this->" serve para nos indicar que estamos utilizando uma variável que pertence à classe que estamos utilizando. Assim o programa não vai confundir uma variável da classe com uma variável que está fora da classe.

Observe que as funções SETs possuem uma variável nas suas "portas de entrada", enquanto que as funções GETs não possuem.

As funções SETs pegam o valor destas variáveis e passam este valor para as variáveis da classe "Usuario_bean".

ATENÇÃO: Mais um detalhe importante. O termo "porta de entrada" não é usualmente utilizado pelos programadores. Quando se falar em "envio de parâmetros para uma função" quer dizer que a função tem variáveis na sua "porta de entrada". Fique atento a isso. Portanto digamos assim: as funções SETs recebem parâmetros enquanto que as funções GETs não recebem.

- o nome das funções GETs iniciam-se com "get" e funções SETs inciam-se com "set". Logo após o get ou set, no início do nome da função, deve vir sempre uma letra maiúscula, por uma questão de consenso entre os programadores.

Vamos agora utilizar a classe Usuario_bean no nosso sistema:

- Abra novamente o arquivo "**cadastro_usuario_exe.php**" que está dentro da pasta "**dgoo**" e faça as seguintes alterações nele:

– Localize neste arquivo as seguintes linhas:

```
echo "<P>Nome do usuário: ".$nome_usuario."<BR>";
echo "E-mail: ".$e_mail_usuario."<BR>";
echo "Telefone: ".$telefone_usuario."</P>";
```

Logo abaixo destas linhas, digite o seguinte:

```
include ('Usuario_bean.php');
$usuarios = new Usuario_bean();
$usuarios->setNome_usuario($nome_usuario);
$usuarios->setEmail_usuario($e_mail_usuario);
$usuarios->setTelefone_usuario($telefone_usuario);
```

Digite um hífen e o sinal de maior, juntos, sem espaço entre eles.

Agora faça o seguinte:

- Na linha onde está escrito

"echo $grava->gravaUsuario($nome_usuario,$e_mail_usuario,$telefone_usuario);"

Altere os parâmetros para

"echo $grava->gravaUsuario($usuarios);"

Salve este arquivo e agora abra o arquivo **Usuario_dao.php**.

Altere o conteúdo da função "gravaUsuario" para o seguinte:

Agora basta passar um único parâmetro)

```
function gravaUsuario( $usuarios ) {
  include ('conexao.php');
$sql="INSERT INTO usuario (nome_usuario, e_mail_usuario, telefone_usuario)
  VALUES ('".$usuarios->getNome_usuario()."','".$usuarios->getEmail_usuario()."','".
            $usuarios->getTelefone_usuario()."')";
  $result = mysql_query($sql);
  daqui para baixo o conteúdo da função continua o mesmo
  if ($result)
        return "Dados cadastrados com sucesso!";
    else
          return "Erro ao tentar cadastrar dados no banco!";
}
```

Não é necessário mudar de linha

- Explicando o código

Incluímos o arquivo da classe "Usuario_bean" para, em seguida podermos criar uma **instância** dessa classe.

A variável **$usuarios** recebeu como valor a instância da classe Usuario_bean.

Em seguida utilizamos a variável "$usuarios" para podermos chamar as funções SETs da classe Usuario_bean, da seguinte forma:

```
$usuarios->setNome_usuario($nome_usuario);
$usuarios->setEmail_usuario($e_mail_usuario);
$usuarios->setTelefone_usuario($telefone_usuario);
```

Desta forma criamos um objeto chamado "$usuarios" que contem todas as variáveis do formulários, com seus respectivos valores. Vamos dizer que, criamos um pacote contendo as variáveis e seus respectivos valores. A este pacote, dar-se o nome de VO (Objeto contendo Valores).

Observe então que, não é necessário mais passar uma série de variáveis (parâmetros) para a função "gravarUsuario()". Basta passar um único parâmetro, que agora é um objeto chamado "usuarios".

Observe que no arquivo "cadastro_usuario_exe.php" foram utilizadas as funções **SETs** da classe "Usuario_bean", enquanto que dentro da função "gravaUsuario" do arquivo "Usuario_dao.php" foram utilizadas as funções **GETs**;

Lembrando que, as funções SETs servem para "setar valores", ou seja, atribuir valores às variáveis, enquanto que as funções GETs servem para entregar estes valores ao programa.

TAREFA um pouco mais difícil:

Agora que fez tudo isso e conseguiu fazer funcionar 100% e entendeu direitinho, então lhe pergunto: - voce é capaz de fazer a mesma coisa no arquivo "altera_usuario_exe.php"?

Lembrando que no arquivo **Usuario_dao.php**, voce vai ter que incluir o conteúdo da função "**alteraUsuario()**". E que voce vai utilizar a classe Usuario_bean que já criamos, além de fazer toda a alteração necessária no arquivo "**altera_usuario_exe.php**".

Não se esqueça de utilizar a seguinte linha de comando:

$usuarios->setId_usuario($id_usuario); no arquivo "altera_usuario_exe.php"

Se conseguir, envie-me mensagem, ou poste no forum, para que eu possa lhe parabenizar pelo seu esforço. Se não conseguir, não se preocupe, mas à frente apresentarei o código completo.

A classe Dao

As classes que operam no banco de dados nós chamamos de classes "DAO". DAO significa *Data Access Object* (ou Objeto de Acesso a dados – Objeto de acesso ao banco de dados).

Voce pode encontrar na internet muitas explicações sobre a classe DAO, não deixe de consultar e estudar a respeito dela e das outras, por exemplo em: http://javafree.uol.com.br/artigo/871452/Introducao-ao-pattern-DAO.html

MVC – Modelo, Visão e Controle

Para auxiliar ainda mais em seu seu estudo sobre Orientação a Objeto, vamos antes falar um pouco sobre MVC.

O que é MVC?

M = Modelo; **V** = Visão e **C** = Controle

Pois bem, além de dividirmos as funções e elementos do nosso sistema em objetos, o ideal é que nosso sistema fique dividido também em módulos (ou camadas) ou pastas. Vamos agrupar (separar) os arquivos por pasta, da seguinte forma: os arquivos que contem as telas do nosso sistema deverão ficar na pasta chamada "Visao", os arquivos referentes à classe bean deverão ficar numa pasta chamada "Modelo" e os arquivos das classes que fazem operações no banco de dados colocaremos também na pasta chamada Modelo.

Dentro da classe Modelo, ainda teremos outras duas pastas: "bean" e "dao".

- Arquivos de tela, na pasta "visao";
- Arquivos das classes bean, na pasta "Modelo/bean";
- Arquivos de banco de dados na pasta "Modelo/dao".

Isso tudo não é só por uma questão de organização, "para ficar bonitinho", essa organização é muito útil, para facilitar na hora que precisar fazer alterações no seu programa, principalmente quando se tratar de um sistema muito grande. Além disso há outros motivos que voce entenderá mais à frente, quando falarmos sobre "regras de negócio" na definição e desenvolvimento de um sistema. Até porque, um programa não se resume apenas a arquivos de visão, arquivos "beans" e arquivos "daos", voce verá mais à frente que é necessário termos outros arquivos, como por exemplo, o arquivo que verifica se os dados informados pelo usuário, no formulário de cadastro, são válidos.

Dentro da pasta "Controle" teremos arquivos que cuidarão da regra de negócio do nosso sistema. Veremos isso mais à frente. Estude mais sobre MVC, existe todo um conceito por trás disso.

TAREFA 1:

Coloque seus arquivos nas respectivas pastas, conforme recomendado à cima, lembrando que, voce terá que fazer uma alteração em todos os arquivos que tiver o comando "include", para que ele consiga localizar os arquivos dentro destas pastas, por exemplo:

Os arquivos **cadastro_usuario.htm** e **cadastro_usuario_exe.php** deverão ficar dentro da pasta "Visao", o arquivo "**Usuario_bean.php**", na pasta "Controle" e o arquivo "**Usuario_dao.php**" na pasta "Modelo".

No arquivo "cadastro_usuario_exe.php", o comando "include" deverá ficar da seguinte forma:

include ('../controle/bean/Usuario_bean.php');

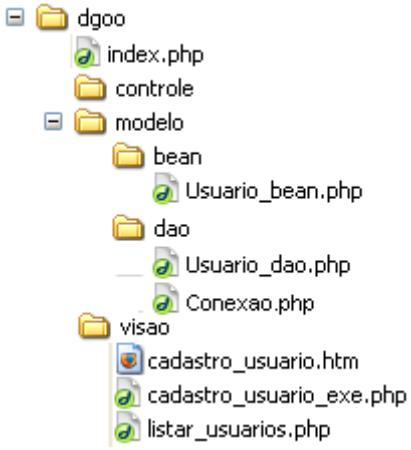
– Observe que "../" serve para informar que o arquivo "Usuario_bean.php" não está dentro da pasta "Visao", mas sim na pasta "Controle/bean", que está fora da pasta "Visao".

Nas empresas que trabalham com desenvolvimento de programas (softwares), é comum que se tenha equipes de trabalho, onde cada uma fique responsável por uma parte do desenvolvimento: uma equipe cria as telas, outra cria as classes de controle e outra as classes de banco de dados (DAO).

Tudo pode ser desenvolvido separadamente e depois de pronto, junta-se todas as partes, formando um único sistema. Para que isso aconteça é necessário

que se faça todo um planejamento sobre como será desenvolvido o sistema. Para estudar sobre projetos de sistemas, pesquise e leia sobre "Engenharia de Software". Muito legal!

A estrutura do nosso sistema deverá ficar da seguinte forma:



Observe que o arquivo index.php ficou fora das pastas. Por enquanto este é o único que deverá ficar aí.

DETALHE: o arquivo de conexão também deverá ficar na pasta dao.

TAREFA 2:

Depois de montar toda esta estrutura, teste novamente o seu sistema.

Se der erro, deve dar nos comandos includes. Faça as alterações necessárias.

PRÓXIMOS PASSOS:

- Iremos criar a classe controleUsuario
- utilizaremos o arquivo cadastro_usuario_exe.php para fazer as inclusões, alterações e exclusões. Ou seja, não teremos mais os arquivos "altera_usuario_exe.php" e nem "excluir_usuario.php".
- Também não teremos mais o arquivo "altera_usuario.php". Utilizaremos um único formulário para fazer inclusões alterações e exclusões.
- Criaremos, em seguida a classe de validação dos dados do formulário.

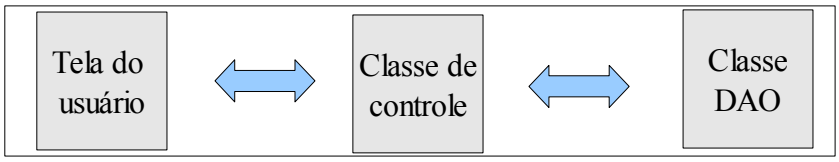
A classe Controle

Como disse anteriormente, esta classe é responsável por toda a regra de negócio do nosso sistema. Entendemos como "Regra de negócio", todas as regras necessárias para o nosso sistema funcionar corretamente, conforme for as necessidades do usuário. Veja abaixo alguns exemplos de regras de negócio que o nosso sistema deverá ter:

- O campo nome de usuário não pode ser gravado em branco e nem com menos de 5 letras;
- O campo e-mail deve ser preenchido com um endereço de e-mail válido. Todo endereço de e-mail deve ter o símbolo "@" e não deve conter espaços em branco.
- O campo telefone não pode ter menos que 8 números.

Antes de desenvolver um sistema, o desenvolvedor deve fazer uma lista contendo todas as regras de negócio que o usuário vai precisar. Não só isso, mas deve ser feito todo um planejamento, conforme ensina a disciplina de Engenharia de Softwares. Todo sistema deve, antes de tudo, ser desenvolvido com qualidade e a única forma de prover qualidade é aplicando metodologias de desenvolvimento, conforme ensina a Engenharia de Software.

A classe controle, além de cuidar das regras de negócio, é também a classe responsável por fazer a ligação entre as telas do sistema e o banco de dados. Ela captura os dados digitados pelo usuário, faz uma validação destes dados e, se tudo estiver correto, chama a classe DAO, para fazer a gravação dos dados.



MVC - **V**ISÃO **C**ONTROLE **M**ODELO

Criando a classe Controle

A classe controle conterá, praticamente o que já temos no arquivo "cadastro_usuario_exe.php", só que de uma forma mais sofisticada.

TAREFA 3:

Digite o seguinte código e grave dentro da pasta "controle", com o nome de "Usuario_controle.php".

```
<?php
        include '../modelo/bean/Usuario_bean.php';
        include '../modelo/dao/Usuario_dao.php';
        include '../modelo/dao/Conexao.php';

    class Usuario_controle {
        protected $nome_usuario;
        protected $e_mail_usuario;
        protected $telefone_usuario;

        protected $usuarios;

        function setUsuario(){
                $this->nome_usuario = $_POST['nome_usuario'];
                $this->e_mail_usuario = $_POST['e_mail_usuario'];
                $this->telefone_usuario = $_POST['telefone_usuario'];

                $this->usuarios = new Usuario_bean();

                $this->usuarios->setNome_usuario($this->nome_usuario);
                $this->usuarios->setEmail_usuario($this->e_mail_usuario);
                $this->usuarios->setTelefone_usuario($this->telefone_usuario);
        }

        function gravaUsuario(){
                $this->setUsuario();
                Conexao::conecta();
                $grava = new Usuario_dao();
                $mensagem = $grava->gravaUsuario($this->usuarios);

                return $mensagem;
        }
} ?>
```

- Explicando o código

Agora a coisa começa a ficar um pouco mais sofisticada e um pouco mais complicada para entender, portanto vai exigir um pouco mais da sua atenção.

Antes de mais nada, devo dizer-lhes o quanto fica mais fácil digitar um código como este utilizando um editor que auxilia na digitação dos códigos PHP. Um exemplo, é o programa EasyEclipseForPhp ou o Eclipse PDT. Ambos são ótimos, mas é necessário ler alguns tutoriais sobre eles para saber utilizá-los.

À medida que voce vai digitando o código ele já vai mostrando o que está digitado errado, ou até complementa automaticamente alguns comandos.

Compare este código com o conteúdo do arquivo "cadastro_usuario_exe.php". Veja que há muitas semelhanças entre eles. Mas, mesmo assim vou explicar algumas coisas que já havia dito antes.

Nas duas primeiras linhas incluímos os dois arquivos que precisaremos utilizar mais adiante.

Para criar uma classe sempre utilizamos o comando "class". E, observe que o nome da classe é o mesmo nome do arquivo "Usuario_controle". Isso é só por uma questão de padronização e de boa prática de programação. Aconselho fazer sempre assim.

Após criarmos a classe, inseri no código as mesmas variáveis que já havia criado na classe Usuario_bean. Bastava então, apenas copiar estas linhas da classe bean e colar neste código.

O termo "protected" que coloquei para cada variável, também é só por uma questão de boa prática de programação. Lendo mais sobre Orientação a Objetos voce entenderá o motivo de utilizar este termo nos nomes das variáveis.

A função setUsuario() contem basicamente a mesma coisa que já tínhamos no arquivo "cadastro_usuario_exe.php".

Para utilizarmos as variáveis dentro da função setUsuario, tivemos que utilizar o "$this", assim estamos indicando para a função, que estas variáveis pertencem à nossa classe Usuario_controle. Se não colocar o "$this" não teremos como passar estes dados para a função "gravarUsuario(), principalmente na variável $usuarios.

Depois que executei todos os comandos "$_POST" criei uma instância da classe Usuario_bean em (`$this->usuarios = new Usuario_bean()`). A partir desta instância utilizei as funções SETs da classe Usuario_bean para criar o pacote de dados do usuário a ser cadastrado. Este pacote ficou armazenado na variável $usuarios.

A função gravaUsuario chama a função setUsuario() para poder pegar todos os dados digitados no formulário. Em seguida ele chama a função de conexão. Na linha seguinte, cria uma instância da classe Usuario_dao e chama a função gravaUsuario da classe DAO, passando como parâmetro a variável $usuarios, que contem todos os dados digitados no formulário.

Entendeu bem isso aí? Este é o fluxo de dados para gravação.

Voce conseguiu ver aí, como a classe controle pega os dados do formulário e envia para o banco de dados?

Agora precisamos transformar o nosso arquivo conexao.php numa classe de conexão. Por enquanto vamos fazer isso da forma mais simples e depois poderemos aprimorá-lo.

TAREFA 4:

O nome do arquivo **conexao.php** agora deverá iniciar-se com letra maiúscula, pois como boa prática de programação, todo arquivo que represente uma classe deverá iniciar-se com letra maiúscula. Faça essa alteração no nome do arquivo.

Observe que no código da classe controle, já colocamos o include da conexão com o nome do arquivo em letra maiúscula.

O conteúdo do arquivo "**Conexao.php**" agora deverá ser o seguinte:

```
<?php

class Conexao {

        static function conecta(){
                $hostname = "localhost";
                $database = "dgestruturado";
                $username = "root";
                $password = "";
                $con = mysql_connect($hostname, $username, $password)
                or die(mysql_error()."Erro ao tentar conectar-se ao banco");

                mysql_select_db($database, $con);
        }
}

?>
```

Observe que o conteúdo continua sendo o mesmo, apenas inclui aí o comando para criação da classe e um comando para criação da função conecta.

Agora temos que fazer uma alteração no arquivo "cadastro_usuario_exe.php"

TAREFA 5:

Abra o arquivo "**cadastro_usuario_exe.php**" e exclua todo o seu conteúdo. Deixe o arquivo em branco.

O conteúdo dele agora (por enquanto) será apenas o seguinte:

```
<?php
  include ('../Controle/Usuario_controle.php');

  $usuarios = new Usuario_controle();

  echo $usuarios->gravaUsuario();

?>
```

Veja que agora o conteúdo deste arquivo ficou bem mais simples. Certo?

Isso quer dizer que toda a complicação do código ficou separada em outro arquivo. Nós chamamos isso de "encapsulamento de código". Leia sobre

"**encapsulamento**" para saber mais a respeito, mas é praticamente isso – separar ou ocultar as partes mais complicadas. Assim facilita ainda mais o entendimento do código e conseqüentemente sua manutenção.

Agora voce já pode testar se o seu programa está gravando todos os dados sem nenhum problema. Para verificar se ele realmente gravou os dados utilize o phpMyAdmin.

Se estiver tudo ok, iremos criar agora a classe de validação dos dados. Mas antes de criarmos esta classe é necessário aprendermos sobre mais um recurso interessante do PHP, que é o "**try.... catch**". Este comando serve para retornar mensagens de erro quando alguma coisa não funcionar bem.

Try... Catch

Observando o código da classe controle, voce deve ter percebido que temos funções que chamam outras funções. Por exemplo a função gravaUsuario() está chamando as funções setUsuario, conecta() e gravaUsuario da DAO.

A função setUsuario, por sua vez, faz uso de outras funções da classe Usuario_bean. Logo, percebe-se que são funções dentro de funções.

O problema aí é o seguinte: quando temos a função 1, chamando a função 2, que por sua vez chama a função 3, se acontecer um erro na função 3, por exemplo, pode ser que este erro nem apareça na sua tela, porque está uma coisa dentro da outra, tudo encapsulado. Para evitar que este erro fique escondido ou para evitar um travamento no sistema devido ao erro ocorrido, criou-se o recurso try... catch.

Este recurso serve para passar para a função anterior, o erro que tiver ocorrido. No caso acima, se ocorresse um erro na função 3, este erro seria passado para a função 2, que por sua vez, passaria para a função 1. Assim o erro poderia ser visualizado na tela, com a mensagem que tiver sido definida pelo programador, e ainda, evitando qualquer travamento no sistema.

Vamos ver um exemplo prático deste recurso na classe de validação dos dados. Será necessário também incluir o try catch na classe de controle e também no arquivo "cadastro_usuario_exe.php".

Criando a classe de validação dos dados

Está é uma classe muito simples, mas muito importante, por contribuir para a garantia da qualidade dos dados que estão sendo cadastrados no banco.

O conteúdo deste arquivo deverá ser baseado nas regras que definimos anteriormente, na página 33.

Voce pode ir melhorando estas regras, à medida que for surgindo novas idéias ou novos campos no seu sistema.

TAREFA 6:

Crie o arquivo "Valida_usuario.php" dentro da pasta controle.

O conteúdo deste arquivo deverá ser o seguinte:

```
<?php
class Valida_usuario {

    function validaUsuario($usuario){

        //verificando o nome
        if (!$usuario->getNome_usuario())
                throw new Exception("É necessário informar o nome do usuário");
                else if (strlen($usuario->getNome_usuario())>5)
                        throw new Exception("O nome do usuário não pode ter menos que 5 letras.");

        //verificando o e-mail
        if ((!strrpos($usuario->getEmail_usuario(),"@"))||(strrpos($usuario->getEmail_usuario()," ")))
                throw new Exception ("O e-mail informado não é válido");

        //verificando o telefone
        if (strlen($usuario->getTelefone_usuario())>8)
                throw new Exception("O número do telefone não pode ter menos que 8 dígitos.");
        }

}
?>
```

- Explicando o código

Aqui estou utilizando comandos IF e ELSE. Voce já deve estar sabendo para que eles servem e como funcionam, não é? Se não souber, pesquise sobre eles, é bem tranquilo. Não é difícil.

Observe que a função "validaUsuario()", recebe como parâmetro o objeto "$usuario", que foi criado lá na classe controle.

A partir deste objeto utiliza-se as funções GETs da classe BEAN, para verificar os valores informados no formulário de cadastro.

São os mesmos GETs que utilizamos para gravar os dados no banco, lá na classe DAO. Lembra?

O primeiro comando IF do código acima, é para testar se o campo nome está em branco. Se tiver em branco, ele retorna uma mensagem de erro, informando que "É necessário informar o nome do usuário".

Nestes comandos "IF", observe que antes da variável "$usuario->getNome_usuario()", temos um sinal de exclamação (!). Este sinal é para indicar que estamos testando se **não** existe um valor na variável em questão.

Em seguida tem um comando "else". Serve para o seguinte: se existir um nome de usuário, então verifique se este nome tem mais que 5 letras. Lembre-se que na regra de negócio que definimos na página 33, ficou especificado que o nome não poderia ter menos que 5 letras.

No teste do endereço de e-mail ele verifica se o endereço digitado no formulário contém o símbolo "@" que deve estar em todo endereço de e-mail. E verifica também se existe espaço em branco no endereço de e-mail, pois não deve existir. Se existir é porque é um endereço inválido.

Depois segue o teste do número do telefone.

Observe que em todos os testes temos o comando "throw new Exception()". É este comando que vai retornar para a função anterior o erro que for encontrado. A função anterior só receberá este erro, se ela tiver um comando

"try.. cath", portanto iremos acrescentar este comando na classe controle, conforme exemplo a seguir. Se não colocarmos o "try...cath", o sistema mostrará um erro e não funcionará.

TAREFA 7:

Implementando o try...catch na classe Usuario_controle

Abra este arquivo e faça as seguinte alterações:

- No início do arquivo, junto aos outros includes, inclua a seguinte linha:

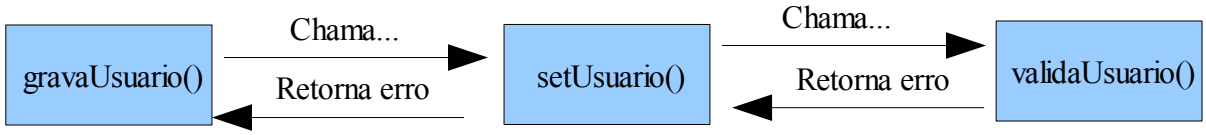
```
include '../controle/Valida_usuario.php';
```

- No final da função setUsuario() (logo abaixo do comando `$this->usuarios->setTelefone_usuario($this->telefone_usuario))`, inclua as seguintes linhas:

```
try{
    Valida_usuario::validaUsuario($this->usuarios);
    }catch (Exception $e){
          throw new Exception ($e->getMessage());
          }
```

O **throw** serve para retornar a mensagem de erro para a função gravaUsuario()

Se a função "setUsuario" está sendo chamada pela função "gravaUsuario", então se ocorrer qualquer erro, a função "Valida_usuario" retornará este erro para a função "setUsuario" e a a função "setUsuario", por sua vez, retornará este erro para a função "gravaUsuario()". O gráfico abaixo representa esse fluxo de mensagens de erro:



Neste caso, se estamos utilizando o "try.. catch" para validar o usuário, então teremos que utilizar também o "try.. catch" dentro da função "gravaUsuario()" da classe controle. Pois, sempre que uma função chamar outra e esta outra utilizar try...catch, a função que está chamando também deverá ter um try...catch. Neste caso deveremos alterar a função grava, da classe Usuario_controle, para:

```
function gravaUsuario(){

      try{
            $this->setUsuario();
      }catch (Exception $e){
            $mensagem = $e->getMessage();
      }

      Conexao::conecta();
      $grava = new Usuario_dao();
      $mensagem = $grava->gravaUsuario($this->usuarios);

      return $mensagem;
}
```

O **catch** serve para pegar a mensagem de erro que vier da função valida_usuario()
- Repare que aqui não se utilizou o **throw**.

LISTANDO OS DADOS UTILIZANDO MVC

Todas as funcionalidades do nosso sistema deverão fazer uso das três camadas (Visão, Controle e Modelo), inclusive a funcionalidade de listagem dos dados.

Uma das regras da boa utilização de MVC é nunca fazer uma ligação direta entre os arquivos da pasta (camada) de visão e os arquivos da pasta (camada) de modelo, a ligação entre estas duas camadas deve ser feita sempre pela camada de controle.

Na tela inicial do nosso sistema, já temos o link para exibir a lista de usuários cadastrados, falta criarmos a função de listagem na camada de controle e na camada de modelo.

Teremos inicialmente duas funções para listagem de dados: uma para listar todos os usuários cadastrados no sistema e outra para exibir/listar apenas um usuário. Esta segunda função vai servir para exibir os dados do usuário no formulário de alteração.

TAREFA 1:

Criar a função que listará todos os usuários cadastrados.

- Abra o arquivo **Usuario_dao.php**
- A função listaUsuarios(), deverá ficar da seguinte forma:

```
function listaUsuarios( ) {

        $sql="SELECT * FROM usuario ORDER BY nome_usuario";
        $result = mysql_query($sql);

        $lista = array();

        while ($rd=mysql_fetch_array($result)){

                $usuario = new Usuario_bean();

                $usuario->setId_usuario($rd['id_usuario']);
                $usuario->setNome_usuario($rd['nome_usuario']);
                $usuario->setEmail_usuario($rd['e_mail_usuario']);
                $usuario->setTelefone_usuario($rd['telefone_usuario']);

                $lista[] = $usuario;
        }

        return $lista;
}
```

Observe a chave do while

Fechando a chave do while.

Os cholchetes servem para acrescentar itens a um array.

- Explicando o código

Observe que no comando SQL tem o "ORDER BY", serve para ordenar a lista de usuários pelo nome.

O comando "mysql_query()" voce já conhece e também o uso de array. Observe que a variável $lista é um array.

O comando “mysql_fetch_array”, também foi explicado na página 18, serve para pegar todos os registros da tabela usuario e transforma-los em um array, para que possam ser exibidos na tela.

O comando **While**, também está explicado na página 19, só que aqui ele está sendo utilizado sem o “do”. Como está explicado lá, este comando serve para percorrer todos os itens de um array. Observe que aqui ele está percorrendo todos os itens do array gerado pelo comando “mysql_fetch_array”, utilizando a variável “$rd”.

O programa fica “rodando em círculo” dentro do While, até conseguir ler todos os registros da tabela usuario. A cada volta que ele dá, ele abre uma instância da classe Usuario_bean(), para executar as funções SETs desta classe, passando como parâmetro, os valores que ele está buscando no banco de dados.

Na prática o comando while faz o seguinte: enquanto a variável $rd estiver recebendo dados do “mysql_fetch_array”, execute os comandos que estiverem entre as chaves do while.

O PHP executa um programa, linha por linha. Assim que ele chega na linha que tem o comando While, ele pega o primeiro ítem do array, que está na variável “$rd” e executa os comandos que estão dentro while, linha por linha. Quando chega no final do While (na chave que fecha o while), ele retorna para o início dos comandos do while e faz tudo novamente pegando o próximo ítem do array.

A cada vez que ele executa o while, o conteúdo da variável “$usuario” é armazenado no array $lista.

Repare na linha de comando: **$lista[] = $usuario**. A cada vez que o while é executado, um registro é acrescentado ao array $lista. Para acrescentar conteúdo a um array é necessário colocar os colchetes na frente da variável array.

O comando “return” passa para a camada de controle a variável $lista, contendo todo o conteúdo da tabela usuario.

Entendeu bem? - Então este é mais um recurso do PHP que voce deve memorizar, ele é muito utilizado em programação.

TAREFA 2:

Criar a função de listagem de usuários na camada de controle.

Abra o arquivo **Usuario_controle** e acrescente a seguinte função:

```
function listaUsuarios(){

    Conexao::conecta();
    $lista = new Usuario_dao();
    $usuarios = $lista->listaUsuarios();

    return $usuarios;
}
```

A variável $usuarios vai receber todo o conteúdo que estiver sendo retornado pela função listaUsuarios lá na classe Usuario_dao().

Cuidado para não colocar esta função do lado de fora da classe Usuario_controle, lembrando que a classe inicia-se com a abertura de uma chave e fecha com outra chave. A função deve estar dentro destas chaves da classe.

TAREFA 3:

- Abra o arquivo **listar_usuarios.php**, que está dentro da pasta "visao".
- Apague todo o seu conteúdo e digite o seguinte:

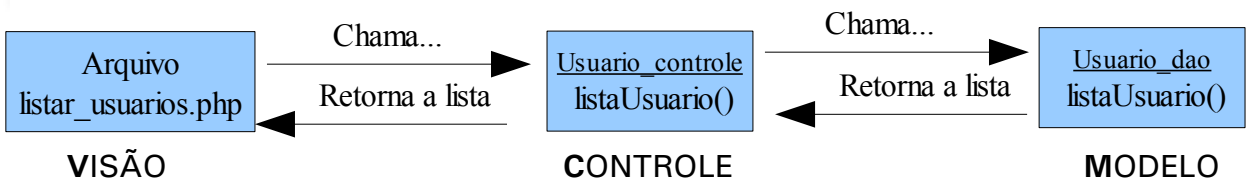
```
<?php
include ('../controle/Usuario_controle.php');

$usuarios = new Usuario_controle();
$lista = $usuarios->listaUsuarios();

?>
<html>
   <body>
     <table border='1'>
        <tr>
           <td>Nome</td>
           <td>E-mail</td>
           <td>Telefone</td>
           <td>Excluir</td>
        </tr>
<?php
foreach ($lista as $r) {
    echo "<tr>";
    echo "<td><a href='cadastro_usuario.php?id_usuario=".$r->getId_usuario()."'>".
      $r->getNome_usuario()."</a></td>";
    echo "<td>".$r->getEmail_usuario()."</td>";
    echo "<td>".$r->getTelefone_usuario()."</td>";
    echo "</tr>";
}?>
     </table>
     <a href="../index.php">voltar</a>
   </body>
</html>
```

- Explicando o código

A variável $lista está recebendo o conteúdo que está sendo passado pela função listaUsuarios() da classe Usuario_controle.

O fluxo dos dados é o seguinte: neste código acima, estamos chamando a função listaUsuarios() da classe Usuario_controle, esta função então, chama a função listaUsuarios() que está na classe Usuario_dao. A função listaUsuarios() da classe Usuario_dao passa a lista de usuários para a função listaUsuarios() da classe Usuario_controle. E a função listaUsuarios() da classe Usuario_controle, por sua vez, repassou esta lista para o arquivo lista_usuarios.php, pois é ele que está chamando a função da classe controle.

O desenho é o mesmo que coloquei para explicar a gravação dos dados:



A novidade aqui neste código é o uso do comando "**FOREACH**". Ele funciona basicamente como o comando "WHILE". Só que ele é uma forma mais simples de percorrer um array. Há casos em que é melhor utilizar o while e outros o foreach. No caso da classe Usuario_dao, é mais comum utilizarmos o while.

TESTE O SISTEMA AGORA

Voce já pode testar se o sistema está listando os dados. Tente resolver os erros que acontecerem, é comum que aconteçam. Se não conseguir peça-me ajuda.

ALTERANDO DADOS UTILIZANDO MVC

Parte 7

Até este ponto já temos toda a parte de cadastro de usuário já funcionando em MVC, com orientação a objetos. Voce verá então, que agora fica bem mais fácil criar as outras partes do sistema, pois muita coisa que já foi criada para o cadastro de usuários, será utilizada para a fazer a alteração e também a exclusão.

Começaremos pela classe "Dao", depois iremos para a "controle" e, por fim, faremos as alterações no formulário de cadastro, para que ele também possa ser utilizado para a alteração de dados. Lembrando que não mais teremos um arquivo separado com o formulário de alteração de dados.

É interessante ter apenas um formulário que sirva para as duas funções, assim evita-se ter que fazer alteração em dois formulários sempre que precisar incluir um novo campo ou fazer uma correção ou uma melhoria qualquer.

TAREFA 1:

Abra o arquivo "Usuario_dao.php" e inclua o seguinte código na função "alteraUsuario()":

```
 function alteraUsuario(Usuario_bean $usuario) {

    $sql="UPDATE usuario  SET
          nome_usuario='".$usuario->getNome_usuario()."',
          e_mail_usuario='".$usuario->getEmail_usuario()."',
          telefone_usuario='".$usuario->getTelefone_usuario()."'
              WHERE id_usuario=".$usuario->getId_usuario();
    $result = mysql_query($sql);

    if ($result)
       echo "Dados alterados com sucesso!";
       else
          echo "Erro ao tentar alterar dados no banco!";
}
```

Por enquanto, para não complicar muito, não utilizaremos aqui o recurso do "try catch". Posteriormente faremos isso, para melhorar ainda mais o nosso sistema.

EXPLICANDO O CÓDIGO:

Detalhe: as cores que vc está vendo aí neste código é a forma como o programa EasyEclipse ou o Eclipse PDT, mostra o código para nós. Ele separa por cores os comandos de PHP, as variáveis e os outros textos.

Observe que o comando SQL que estamos utilizando aí é o mesmo que foi utilizado anteriormente (lá na página 22). A única diferença está nas variáveis: ao invés de utilizamos $nome_usuario", por exemplo, estamos utilizando as funções GETs para pegar os valores do objeto. Releia a explicação sobre a gravação dos dados com Orientação a Objetos (página 31), para relembrar alguns conceitos que também se aplicam a este caso.

O detalhe importante do comando UPDATE do SQL é sempre utilizar o WHERE indicando em qual registro deverá ser feito a alteração. Se não colocar isso, a alteração vai ocorrer em todos os registros do seu banco de dados.

TAREFA 2:

Ok, agora vamos para a classe "Usuario_controle".

Basta, simplesmente selecionar e copiar a função gravaUsuario(), depois colar nas linhas abaixo e alterar o palavra "grava" para "altera". Veja o código abaixo:

```
function alteraUsuario(){

            $this->setUsuario();

            Conexao::conecta();
            $grava = new Usuario_dao();
            $mensagem = $grava->alteraUsuario($this->usuarios);

            return $mensagem;
      }
```

Observe como o código é identico à função gravaUsuario, só mudando a palavra "grava", para "altera".

TAREFA 5:

Ok, Abra o arquivo "**cadastro_usuario_exe.php**" faça a seguinte alteração:

- Apague a seguinte linha:

```
echo $usuarios->gravaUsuario();
```

- Inclua as seguintes linhas:

```
if ($_POST['Submit']=="Cadastrar")
          echo $usuarios->gravaUsuario();

if ($_POST['Submit']=="Alterar")
          echo $usuarios->alteraUsuario();
```

EXPLICANDO O CÓDIGO:

O formulário de cadastro, que agora servirá também para alteração, terá dois botões: um para gravar e outro para alterar. Por isso o código acima vai chamar a função de "gravar" quando voce clicar no botão para gravar, e chamará a função "alterar", se voce clicar no botão de alteração. Mostrarei isso a seguir na alteração que faremos no arquivo **cadastro_usuario.htm**.

TAREFA 6:

Renomeie o arquivo "cadastro_usuario.**htm**" para "cadastro_usuario.**php**", ou seja, altere apenas a extensão do nome do arquivo. Ele agora vai passar a trabalhar com variáveis do PHP, portanto não poder ser mais um simples arquivo html.

O que devo desenvolver primeiro?

Se for começar a desenvolver do ponto zero, recomendo que siga a seguinte sequência:

- Primeiro criar o banco de dados, contendo todas as tabelas e campos que o sistema fará uso. Isso só é possível fazer se já tiver tudo bem definido no planejamento do seu sistema. É necessário estudar modelagem de dados, para fazer esta atividade bem feita.

- segundo passo: criar as classes BEANs, contendo todas as variáveis que o sistema fará uso e suas respectivas funções GETs e SETs.

Para saber quais serão todas as variáveis que farei uso, preciso antes ter isso já descrito no planejamento do sistema, baseando-se principalmente nos formulários que o sistema deverá ter.

Se, por exemplo estou desenvolvendo um sistema de vendas, com certeza meu sistema terá cadastro de clientes e de produtos. Terei então que desenvolver uma classe BEAN para clientes e outra classe BEAN para produtos.

- Em seguida crio as classes DAOs, contendo todas as funções necessárias para gravar, alterar, listar e excluir dados. Também fazendo uma classe para clientes e outra para produtos.

- Depois crio as duas classes de controles também, contendo todas as regras que o sistema deverá atender e fazendo todas as ligações necessárias entre as telas do sistema e a classe DAO. Incluindo aqui também a classe de validação dos dados. Lembrando que, a classe de controle serve para pegar os dados que vierem das telas, validar estes dados e depois enviar para o banco de dados.

- As telas podem até serem desenvolvidas por uma outra pessoa, ao mesmo tempo em que se está programando estas outras partes do sistema, mas, se for voce mesmo que criará as telas, sugiro que crie depois que já tiver desenvolvido a classe DAO. Assim, voce já poderá ir testando algumas coisas, como por exemplo, os menus dinâmicos que o seu sistema poderá ter. Menus dinâmicos são menus (combos) que buscam informações no banco de dados para poderem mostrar as opções que o usuário poderá selecionar. Talvez eu mostre mais à frente como se faz esse tipo de combo. Pesquisando sobre combos dinâmicos, voce encontra muito código pronto na internet.

OK, a apostila ainda não está completa. Aguarde as próximas edições, em breve.