Aula 8º

Estudo de Caso

Olá,

Esta é a última aula da disciplina Desenvolvimento Voltada à Web II. Na aula anterior, foram utilizados os conceitos de MVC para exemplificar como separar o desenvolvimento entre a parte visual que interage com o usuário (View), a que manipula os dados (Model) e a de controle, que integra as duas (Controller).

Para encerrar nossa disciplina, nesta aula, em uma única seção, faremos o estudo de caso com um banco de dados de exemplo. Será implementado no modelo MVC o CRUD, para a tabela Cidades.

Leia atentamente esta aula e, se tiver alguma dúvida, use os recursos que estão na sua área do aluno.

Bons estudos!

Objetivos de aprendizagem

Ao término desta aula, vocês serão capazes de:

- desenvolver o CRUD para uma tabela com o MVC.

Seções de estudo

1 – Tabela Cidade

1- Tabela Cidade

Os arquivos utilizados para desenvolver este estudo de caso estarão disponíveis nos materiais de aula, porém, para um estudo mais detalhado, indicamos que você reimplemente os arquivos para uma melhor compreensão.

Utilizaremos o banco de dados da Figura 1 como base para a construção de uma interface que o usuário pode Ler, Inserir, Atualizar e Deletar informações da tabela Cidade.

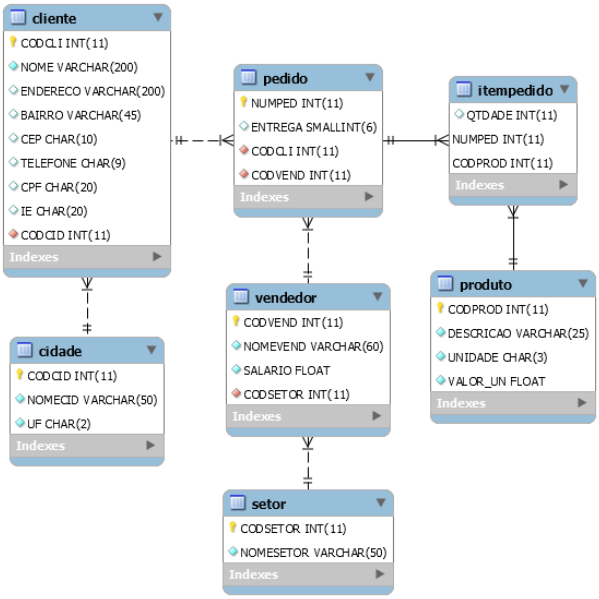

Figura 1 – Estrutura BD. Fonte: Acervo pessoal.

A tabela Cidade contém dois campos que podem ser editados (NOMECID e UF), e o campo CODCID, que é a chave primária dessa tabela. Então, o usuário deve conseguir visualizar uma tabela com o nome da cidade e o estado, e ter botões para atualizar e excluir, além de campos de texto para a entrada de novas cidades.

Para o modelo MVC, seguiremos a base criada na Aula 07, acrescentando um arquivo de controle, que ficará responsável por efetuar as ações solicitadas pelo usuário.

```
mvc
├── model/
│   └── Cidade.php
├── view/
│   └── cidade_view.php
├── controller/
│   ├── CidadeController.php
│   └── EditarController.php
└── index.php
```

Figura 2 – Estrutura de arquivos MCV. Fonte: Acervo pessoal.

O arquivo index, na raiz do projeto, faz referência e chamada ao controlador, que constrói a página através do método lista() do objeto instanciado de CidadeController().

```
<?php
    /*
    Programa:   index.php
      Autor:          Felipe Perez
      Data:           30/10/2019
    */
    require_once          'controller/
CidadeController.php';

    $obj = new CidadeController();
    $obj->listar();

?>
```

O CidadeController, presente no arquivo CidadeController.php, na pasta controller, é quem se comunica com o modelo. Ele solicita que todas as cidades sejam retornadas pelo modelo. Após isso, ele encaminha essas cidades, que são retornadas, para a visão que irá exibir os dados retornados.

```
<?php
    /*
    Programa:  c o n t r o l l e r /
CidadeController.php
      Autor:          Felipe Perez
      Data:           30/10/2019
    */
    require_once "model/Cidade.php";

    class CidadeController{
        public function listar(){
            $cidade = new Cidade();
            $cidades   =   $cidade-
>read();

            $_REQUEST["cidades"] =
$cidades;

            require_once    "view/
cidade_view.php";
        }
    }
?>
```

No modelo, os gets e sets são implementados para manipular os atributos privados que são criados na classe Cidade. Além disso, a conexão com o banco de dados é realizada através do método connectaBD(), que é chamado no construtor do objeto. Então, quando esse objeto é criado, a conexão com o banco de dados já é fechada e pode ser utilizada através do atributo conn.

O controlador solicita, então, através do método read(), no código acima, que no modelo executa uma query no banco de dados, para que todos as cidades sejam exibidas. Coloca, então, todas as cidades em um array, uma a uma (pelo método push()), e retorna para o controlador.

```
<?php
    /*
```

```
      Programa:   model/Cidade.php
        Autor:          Felipe Perez
        Data:           30/10/2019
      */

   class Cidade{
      private $codcid;
      private $nomecid;
      private $uf;
        private $conn;

        function __construct() {
            $this->connectaBD();
        }

        public function getCodCid(){
            return $this->codcid;
        }
      public function setCodCid($codcid)
{
            $this->codcid=$codcid;
        }

        public function getNomeCid(){
            return $this->nomecid;
        }
     public function setNomeCid($nomecid)
{
            $this->nomecid=$nomecid;
        }

        public function getUF(){
            return $this->uf;
        }
        public function setUF($uf){
            $this->uf=$uf;
        }
        private function connectaBD(){
            $server = "localhost";
            $user = "root";
           $pass = ""; $mydb = "vendas";
         $this->conn = new mysqli($server,
$user, $pass, $mydb);
                if($this->conn->connect_
error){
                    die("Conexão Falhou:
".$conn->connect_error);
            }
        }

        public function create(){
       $sql = "INSERT INTO cidade (NOMECID,
UF)  VALUES  ('".$this->getNomeCid()."',
'".$this->getUF()."')";
            $this->conn->query($sql);
        }

     public function update(){
             $sql = "UPDATE cidade SET
NOMECID='".$this->getNomeCid()."',
UF='".$this->getUF()."'           WHERE
CODCID=".$this->getCodCid();
            $this->conn->query($sql);
        }

        public function delete(){
             $sql = "DELETE FROM cidade
WHERE CODCID=".$this->getCodCid();
            $this->conn->query($sql);
        }

        public function read(){
           $sql = "SELECT * FROM cidade";
            $returnValue = array();

                $result  =  $this->conn-
>query($sql);

            if ($result != null) {
                  while($row = $result-
>fetch_array(MYSQLI_ASSOC)){
                    if (!empty($row)) {
                                    array_
push($returnValue,$row);
                    }
                }
            }

            return $returnValue;
        }

   }
```

E fica, então, como responsabilidade da Visão da cidade, por apresentar a tabela com as cidades que são recebidas através da variável $_REQUEST.

```
   <?php
      /*
      Programa:   view/cidade_view.php
        Autor:          Felipe Perez
        Data:           30/10/2019
      */

      $cidades = $_REQUEST['cidades'];
   ?>
   <!DOCTYPE html>
   <html lang="pt-br">
      <head>
            <title>Cidade MVC</title>
      </head>
      <body>
```

Dentro do arquivo de visão é criado um formulário contendo três campos de texto, nome da cidade, uf e codcid, porém, esse último fica oculto, sendo utilizado somente quando uma cidade for excluída ou alterada. São criados, dentro do formulário, também dois botões, um para efetuar a ação (seja criar ou alterar) e outro para resetar o formulário.

```
                <form    method="post"
action="controller/EditarController.
php" id="form" name="form" >
                    <input type="text"
id="nomecid"  name="nomecid"  value=""
placeholder="NomeCid" required>
              <input type="text" id="uf"
name="uf"   value=""   placeholder="UF"
required>
                   <input type="hidden"
id="codcid" name="codcid" value="">
                 <button type="submit"
id="btsubmit"           name="btsubmit"
value="criar">Criar</button>
             <button onclick="btReset()"
type="reset"      value="reset">Reset</
button>
           </form>
```

A tabela tem no cabeçalho o nome das colunas (CodCid, NomeCid e UF), e incluímos duas colunas para representar os botões de edição e exclusão. As cidades são incluídas na tabela utilizando o foreach, que percorre o array cidades. Os botões de edição e exclusão são caracteres que, ao serem clicados, realizam a função que está descrita no atributo onclick (edit ou remove).

```
           <table>
                <tr>
                     <th>CodCid</th>
                  <th>NomeCid</th>
                  <th>UF</th>
                  <th>Editar</th>
                  <th>Excluir</th>
               </tr>
          <?php  foreach  ($cidades  as
$cidade): ?>
              <tr>
                       <td><?php  echo
$cidade["CODCID"]; ?></td>
                  <td id="nomecid_<?php
echo  $cidade["CODCID"];  ?>"><?php  echo
$cidade["NOMECID"]; ?></td>
                      <td  id="uf_<?php
echo  $cidade["CODCID"];  ?>"><?php  echo
$cidade["UF"]; ?></td>
           <td onclick="edit('<?php echo
$cidade["CODCID"];  ?>')"  style="text-
align: center;cursor: pointer;">O</td>
             <td onclick="remove('<?php
echo      $cidade["CODCID"];      ?>')"
style="text-align:       center;cursor:
pointer;">X</td>
               </tr>
       <?php endforeach; ?>
           </table>
```

Por fim, neste arquivo, um código JavaScript foi implementado. Ele possui as funções de edição e remoção. Quando o botão de editar é clicado, o JavaScript preenche os campos de texto do formulário com os dados que já estão salvos no banco de dados. Além disso, altera o texto e valor do botão que deverá ser clicado, para alterar os dados.

O botão de remover efetua basicamente a mesma função do editar. A função que será executada é armazenada no botão com id "btSubmit". Por padrão ele tem o valor criar, e pode ser alterado para editar ou remover.

O funcionamento do botão de reset também é implementado. Ele limpa os campos de texto do formulário e retorna o botão "btSubmit" para o valor e texto criar.

```
         <script  language="javascript"
type="text/javascript">
               function edit(id) {
                                document.
getElementById("nomecid").value
=                               document.
getElementById("nomecid_"+id).
innerHTML;
                                document.
getElementById("uf").value  =  document.
getElementById("uf_"+id).innerHTML;
                                document.
getElementById("btsubmit").value      =
"editar";
                                document.
getElementById("btsubmit").innerHTML  =
"Editar";
                                document.
getElementById("codcid").value = id;
               }
               function remove(id) {
                                document.
getElementById("nomecid").value
=                               document.
getElementById("nomecid_"+id).
innerHTML;
                                document.
getElementById("uf").value  =  document.
getElementById("uf_"+id).innerHTML;
                                document.
getElementById("btsubmit").value      =
"remover";
                                document.
getElementById("btsubmit").innerHTML  =
"Remover";
                                document.
getElementById("codcid").value = id;
                                document.
getElementById("btsubmit").click();
               }
               function btReset(){
                                document.
getElementById("codcid").value = "";
                                document.
getElementById("btsubmit").value      =
"criar";
                                document.
getElementById("btsubmit").innerHTML  =
```

```
"Criar";
                }
            </script>
        </body>
    </html>
```

A Figura 3 apresenta a interface que será exibida para o usuário criada pela visão. Por padrão, a página inicia com a inserção de novas cidades Quando o caractere 'O', na coluna editar, for clicado no conteúdo da linha da tabela, irá preencher as caixas de texto e o texto do botão será mudado para editar como na Figura 4.

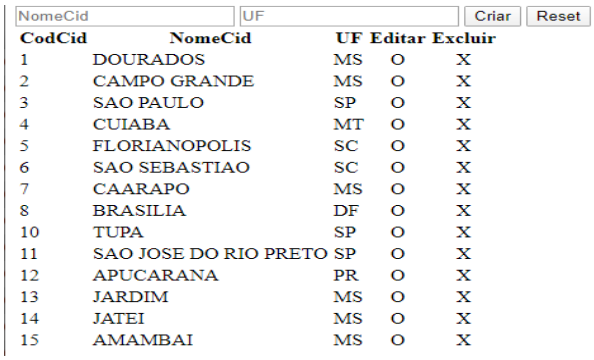

Figura 3 – Criar Cidade. Fonte: Acervo pessoal.

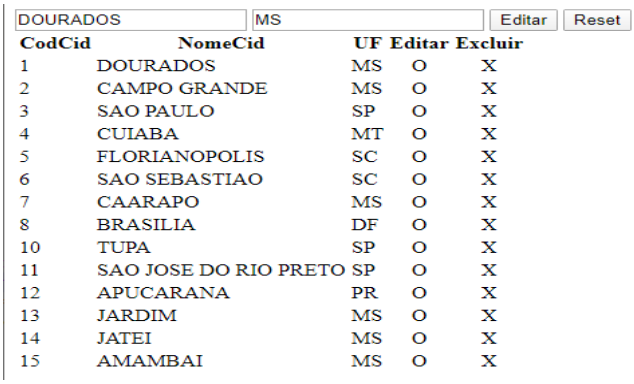

Figura 4 – Editar Cidade. Fonte: Acervo pessoal.

A ação do formulário é enviar os dados através do método POST, para a página EditarController.php da pasta controller. Serão enviados os seguintes dados:

ID	Conteúdo	Função
codcid	"1", "2", "3"	Envia o código da cidade quando a ação for editar ou remover
nomecid	"Dourados", "Campo Grande"	Envia o Nome da Cidade
uf	"MS", "SP"	Envia o Estado da Cidade
btsubmit	"criar", "editar", "remover"	Enviar o tipo de ação a ser realizada

O controlador recebe, então, os dados, e através do método update(), que recebe por parâmetro cada um dos dados da tabela acima, efetua a operação utilizando o modelo que foi implementado para a cidade (create(), update(), delete()). Uma classe do modelo é criada e são setados os valores que irão ser persistidos no banco de dados.

Após a execução dos métodos, a função **header("Location: ../index.php")** é executada, chamando novamente o index.php, que irá listar novamente as cidades acrescentando, atualizando ou removendo conforme a ação que foi chamada.

```
<?php
    /*
    Programa: controller/
EditarController.php
     Autor:          Felipe Perez
     Data:           30/10/2019
     */

      require_once "../model/Cidade.
php";

    class EditarController{
        private $cidade;

        public function __construct()
{
                $this->cidade = new
Cidade();
        }

                public function
update($nomecid,$uf,$codcid,$submit){
                    $this->cidade-
>setNomeCid($nomecid);
            $this->cidade->setUF($uf);
              switch ($submit) {
                  case 'criar':
                      $this->cidade-
>create();
                  break;
                  case 'editar':
                      $this->cidade-
>setCodCid($codcid);
                      $this->cidade-
>update();
                  break;
                  case 'remover':
                      $this->cidade-
>setCodCid($codcid);
                      $this->cidade-
>delete();
                  break;
              }
                 header("Location: ../
index.php");
              exit();
          }
      }

     $editar = new EditarController();
      if(isset($_POST['btsubmit'])){
                  $editar->update($_
POST['nomecid'],$_POST['uf'],$_
POST['codcid'],$_POST['btsubmit']);
      }
```

Retomando a aula

Chegamos ao final da oitava aula. Vamos recordar?

1 – Tabela Cidade

Nesta seção, foi demonstrado um estudo de caso para a inclusão, alteração e remoção de cidades em uma tabela do banco de dados. Utilizando os conceitos aprendidos de MVC e conexão com o BD, foi possível criar o modelo da cidade, os controladores que manipulam as informações e as visões que apresentam as informações para o usuário. Ao final, a estrutura completa para a criação do estudo de caso foi apresentada.

Vale a pena

Vale a pena **ler**

SOARES, W. *Programação Web com PHP 5,* Érica, 2014.

BEIGHLEY, L; MORRISON, M. *Use a Cabeça!* PHP & MySQL. Alta Books, 2010.

SILVA, J. M. C. *PHP na prática 200 Exercícios resolvidos.* Elsevier, 2014.

BENTO, E. J. *Desenvolvimento web com PHP e MySQL.* Editora Casa do Código, 2014.

Vale a pena **acessar**

WELENIR, W. In: *Conceito de MVC e sua funcionalidade usando o PHP,* 2011. Disponível em: https://www.devmedia.com.br/conceito-de-mvc-e-sua-funcionalidade-usando-o-php/22324. Acesso em: 30 out. 2019.

FIGUEIREDO, E. In: *Entendendo o padrão MVC na prática*, 2015. Disponível em: https://tableless.com.br/entendendo-o-padrao-mvc-na-pratica/. Acesso em: 30 out. 2019.

RAMOS, A. In: *O que é MVC?*, 2015. Disponível em: https://tableless.com.br/mvc-afinal-e-o-que/. Acesso em: 30 out. 2019.

GONÇALVES, J. In: *Construindo um simples framework MVC com PHP*, 2019. Disponível em: https://medium.com/@jardelgoncalves1996/construindo-um-simples-framework-mvc-com-php-349e9cacbeb1. Acesso em: 30 out. 2019.

Referências

Alura. In: HTTP: *Diferenças entre GET e POST*, 2019. Disponível em: https://www.alura.com.br/artigos/diferencas-entre-get-e-post. Acesso em: 22 out. 2019.

Asort. In: *PHP Documentation*, asort, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.asort.php. Acesso em: 25 out. 2019.

BEIGHLEY, L; MORRISON, M. *Use a Cabeça!* PHP & MySQL. Alta Books, 2010.

BENTO, E. J. *Desenvolvimento web com PHP e MySQL.* Editora Casa do Código, 2014.

Combine. In: *PHP Documentation,* array_combine, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-combine.php. Acesso em: 25 out. 2019.

Date. In: *PHP Documentation*, date, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.date.php. Acesso em: 25 out. 2019.

Decode. In: *PHP Documentation,* json_decode, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.json-decode.php. Acesso em: 25 out. 2019.

Diff. In: *PHP Documentation*, array_diff, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-diff.php. Acesso em: 25 out. 2019.

Diff. In: *PHP Documentation,* date_diff, 2019. Disponível em: https://www.php.net/manual/pt_BR/datetime.diff.php. Acesso em: 25 out. 2019.

Encode. In: *PHP Documentation*, json_encode, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.json-encode.php. Acesso em: 25 out. 2019.

FIGUEIREDO, E. In: *Entendendo o padrão MVC na prática*, 2015. Disponível em: https://tableless.com.br/entendendo-o-padrao-mvc-na-pratica/. Acesso em: 30 out. 2019.

GAMA, A. In: *O que é JSON*, 2011. Disponível em: https://www.devmedia.com.br/o-que-e-json/23166. Acesso em: 25 out. 2019.

GONÇALVES, J. In: *Construindo um simples framework MVC com PHP*, 2019. Disponível em: https://medium.com/@jardelgoncalves1996/construindo-um-simples-framework-mvc-com-php-349e9cacbeb1. Acesso em: 30 out. 2019.

IMC. In: *Cálculo IMC*, 2019. Disponível em: https://www.programasaudefacil.com.br/calculadora-de-imc. Acesso em: 22 out. 2019.

In_array. In: PHP Documentation, in_array, 2019. Disponível em: <https://www.php.net/manual/pt_BR/function.in-array.php>. Acesso em: 25 out. 2019.

KeyExists. In: *PHP Documentation*, array_key_exists, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-key-exists.php. Acesso em: 25 out. 2019.

Ksort. In: *PHP Documentation*, ksort, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.ksort.php. Acesso em: 25 out. 2019.

Map. In: *PHP Documentation,* array_map, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-map.php. Acesso em: 25 out. 2019.

Merge. In: *PHP Documentation*, array_merge, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-merge.php. Acesso em: 25 out. 2019.

PHP. In: *PHP Documentation*, 2019. Disponível em: https://www.php.net/docs.php. Acesso em: 20 out. 2019.

PIRES, J. In: *O que é API? REST e RESTful? Conheça as definições e diferenças*, 2017. Disponível em: https://becode.com.br/o-que-e-api-rest-e-restful/. Acesso em 26 out. 2019.

Pop. In: *PHP Documentation,* array_pop, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-pop.php. Acesso em: 25 out. 2019.

Push. In: *PHP Documentation,* array_push, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-push.php. Acesso em: 25 out. 2019.

RAMOS, A. In: *O que é MVC?,* 2015. Disponível em: https://tableless.com.br/mvc-afinal-e-o-que/. Acesso em: 30 out. 2019.

Search. In: *PHP Documentation*, array_search, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.array-search.php. Acesso em: 25 out. 2019.

SILVA, J. M. C. *PHP na prática 200 Exercícios resolvidos.* Elsevier, 2014.

SOARES, E. In: *O PHP*: If/else e o operador ternário, 2017. Disponível em: https://www.devmedia.com.br/php-if-else-e-o-operador-ternario/38219. Acesso em 26 out. 2019.

SOARES, W. *Programação Web com PHP 5*, Érica, 2014.

TEO. In: *HERANÇA E POLIMORFISMO EM PHP,* 2019. Disponível em: https://www.todoespacoonline.com/w/2014/08/heranca-e-polimorfismo-em-php-orientado-objetos/. Acesso em: 23 out. 2019.

Timezone. In: *PHP Documentation*, date_default_timezone_set, 2019. Disponível em: https://www.php.net/manual/pt_BR/function.date-default-timezone-set.php. Acesso em: 25 out. 2019.

W3SCHOOLS. In: *PHP - What is OOP?*, 2019. Disponível em: https://www.w3schools.com/php/php_oop_what_is.asp. Acesso em: 23 out. 2019.

W3SCHOOLS. In: *PHP File Open/Read/Close,* 2019. Disponível em: https://www.w3schools.com/php/php_file_open.asp. Acesso em: 21 out. 2019.

W3SCHOOLS. In: *PHP Form Handling*, 2019. Disponível em: https://www.w3schools.com/php/php_forms.asp. Acesso em: 22 out. 2019.

WELENIR, W. In: *Conceito de MVC e sua funcionalidade usando o PHP*, 2011. Disponível em: https://www.devmedia.com.br/conceito-de-mvc-e-sua-funcionalidade-usando-o-php/22324. Acesso em: 30 out. 2019.

Minhas anotações