Módulo 3 – Do estruturado à Orientação a Objetos

Neste workbook do módulo 3, revisaremos os principais conceitos vistos nas videoaulas.

Conexão manual à base de dados

Uma conexão manual e procedural pode utilizar-se de funções tais como: mysqli_connect, pg_connect, e outras dependendo do banco de dados.

```
<?php
$conn = mysqli_connect('127.0.0.1', 'root', 'mysql', 'livro');
$query = 'SELECT codigo, nome FROM famosos';
$result = mysqli_query($conn, $query);
if ($result) {
   while ($row = mysqli_fetch_assoc($result)) {
      echo $row['codigo'] . ' - ' . $row['nome'] . "<br>\n";
   }
}
mysqli_close($conn);
```

Conexão PDO à base de dados

Como realizar uma conexão com a biblioteca PDO. A biblioteca PDO abstrai o acesso, sendo que somente a linha passada para o construtor muda de um banco para outro.

```
<?php
try {
   $conn = new PDO('pgsql:dbname=livro;user=postgres;password=;host=localhost');
   $result = $conn->query("SELECT codigo, nome FROM famosos");
   if ($result) {
      foreach($result as $row) {
         // exibe os resultados
         echo $row['codigo'] . ' - ' . $row['nome'] . "<br>\n";
      }
   }
   $conn = null;
}
catch (PDOException $e) {
   print "Erro!: " . $e->getMessage() . "<br>";
}
```

Nível 1 - Programação procedural, um script por ação

O programa ainda é bastante básico, estruturado, com PHP, HTML e SQL misturados, onde cada ação (incluir, editar, excluir, listar) é um script separado.

nivel1/pessoa_save_insert.php

```
<?php
$dados = $_POST;
$dsn = "host=localhost port=5432 dbname=livro user=postgres password=";
$conn = pg_connect($dsn);
```

Nível 2 - Agrupando ações comuns em scripts

Reunimos ações comuns em torno de menos scripts (incluir junto com editar, listar junto com excluir), e passamos a trabalhar com o conceito de script e ação realizada.

nivel2/pessoa_form.php

```
<?php
   if (!empty($_REQUEST['action'])) {

     if ($_REQUEST['action'] == 'edit') {
        $id        = (int) $_GET['id'];
        $result    = pg_query($conn, "SELECT * FROM pessoa WHERE id='{$id}'");
```

Nível 3 - Separando o HTML com micro templates

Trabalhamos com o HTML em arquivos a parte, chamados de templates. Com isso, separamos o visual do resto do programa.

nivel3/pessoa_form.php

```
<?php

...

$form = file_get_contents('html/form.html');
$form = str_replace('{id}',       $pessoa['id'],       $form);
$form = str_replace('{nome}',     $pessoa['nome'],     $form);
```

Nível 4 - Separando o acesso a dados com funções

Separamos todas as funções que mexem com banco de dados em um arquivo (ainda estruturado), contendo uma série de funções (insere_pessoa, exclui_pessoa). Separamos as funções de persistência do programa principal.

nivel4/pessoa_form.php

```
<?php
require_once 'db/pessoa_db.php';

if (!empty($_REQUEST['action'])) {
    if ($_REQUEST['action'] == 'save') {
        $pessoa = $_POST;
        if (empty($_POST['id'])) {
            $result = insert_pessoa($pessoa);
        }
        else {
            $result = update_pessoa($pessoa);
        }
    }
```

Nível 5 - Separando o acesso a dados com classes

Transformamos as funções de acesso à base de dados em classes de modelo com métodos de manipulação de dados. Introduzimos a biblioteca PDO e começamos a trabalhar com exceções. Nossa camada de banco já está utilizando classes.

nivel5/pessoa_form.php

```
<?php

if (!empty($_REQUEST['action'])) {
    try {
        if ($_REQUEST['action'] == 'edit') {
            $id     = (int) $_GET['id'];
            $pessoa = Pessoa::find($id);
        }
        else if ($_REQUEST['action'] == 'save') {
            $pessoa = $_POST;
            Pessoa::save($pessoa);
        }
    }
```

Nível 6 - Melhorando as conexões e a segurança

Refatoramos nossa classe de banco de dados para esta não apresentar mais os dados de conexão (banco, usuário, e senha) explicitamente no código, e aumentamos a segurança ao trabalhar com prepared statements para evitar ataques como SQL injection.

nivel6/classes/Pessoa.php

```
<?php
class Pessoa {
   private static $conn;

   public static function save($pessoa) {
      $conn = self::getConnection();

      if (empty($pessoa['id'])) {

         $sql = "INSERT INTO pessoa (id, nome, endereco, bairro,  telefone, email, id_cidade)
                     VALUES ( :id, :nome, :endereco, :bairro, :telefone, :email, :id_cidade )";
      }
```

Nível 7 - Transformando páginas em classes de controle

Refatoramos nossos programas principais, que ainda são estruturados, e transformamos estes em classes de controle (Ex: PessoaForm, PessoaList), onde cada ação do usuário é identificada por um método. Também alteramos o fluxo da aplicação, e agora temos um único arquivo (index.php) por onde passam todas as rotas.

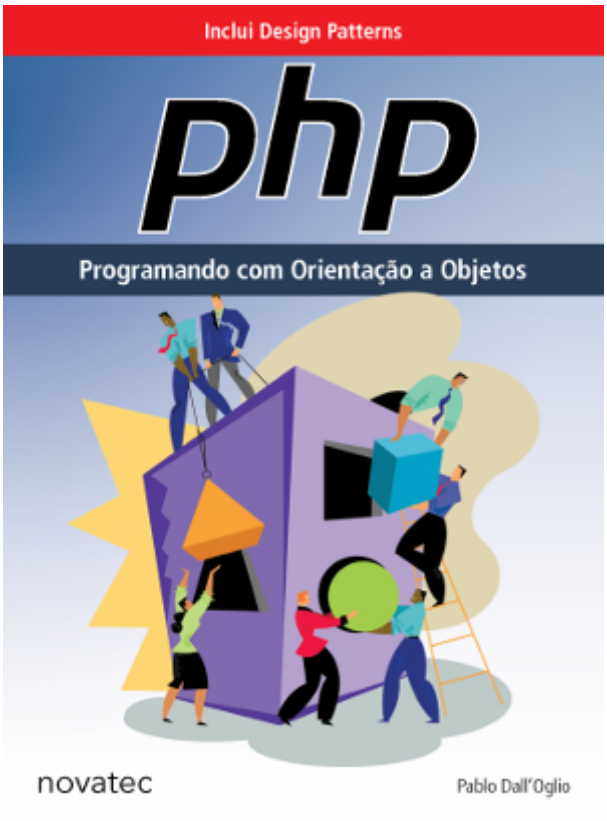
nivel7/PessoaForm.php

```
<?php
require_once 'classes/Pessoa.php';
require_once 'classes/Cidade.php';

class PessoaForm {

   public function edit($param) {
      try  {
         $id     = (int) $param['id'];
         $pessoa = Pessoa::find($id);
         $this->data = $pessoa;
      }
      catch (Exception $e) {
         print $e->getMessage();
      }
   }
```

Livro PHP Programando com Orientação a Objetos



www.adianti.com.br/phpoo