PHP Avançado

VERSÁTIL
tecnologia em informação

SUMÁRIO

SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

PHP Avançado

1. Classes e Objetos

A classe é uma estrutura ou esqueleto que define um conjunto de características similares. Uma **determinada classe define o comportamento de seus objetos** usando métodos e modificando seus estados com os atributos.

Os atributos são as características de um objeto, vamos a um exemplo prático, imagine que temos um objeto chamado "Pessoa" quais seriam as características desse objeto? Cor dos olhos, altura, peso, cor da pele, logo entende-se que atributos são essas informações pertinentes a um determinado objeto. Os métodos são as ações de um objeto, ou seja aquilo que eles executam durante uma determinada solicitação, usando a mesma analogia do objeto "Pessoa", imagine as ações de uma pessoa "andar", "olhar", "pensar" e assim sabemos o que ele vai fazer conforme a solicitação.

Vamos criar agora todas as partes do projeto, seguindo os arquivos e seus códigos abaixo:

Index.php

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Documento sem título</title>
6 </head>
7
8 <body>
9
10 <form method="post" action="ope.php" id="formlogin" name="formlogin" >
11 <fieldset id="file">
12 <legend>LOGIN</legend><br />
13 <label>NOME : </label>
14
15 <!-- o campo "name" dentro do input é importante, pois será ele que
16 armazenará os dados digitados . -->
17
18 <input type="text" name="login" id="login" /><br />
19 <label>SENHA :</label>
20 <input type="password" name="senha" id="senha" /><br />
21 <input type="submit" value="LOGAR " />
22 </fieldset>
23 </form>
24 </body>
25 </html>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

ope.php

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Documento sem título</title>
6 </head>
7
8 <body>
9 <?php
10 // session_start inicia a sessão
11 session_start();
12 // as variáveis login e senha recebem os dados digitados na
página anterior
13 $login = $_POST['login'];
14 $senha = $_POST['senha'];
15 // Verifica se o login e a senha digitados batem com os valores
fornecidos
16 if($login == "admin" && $senha == "123" )
17 {
18 // coloca na sessão o login e a senha caso tenham sido os
corretos e redireciona
19 //para a url site.php
20 $_SESSION['login'] = $login;
21 $_SESSION['senha'] = $senha;
22 header('location:site.php');
23 }
24 else{
25 //caso login ou senha estejam incorretos são destruídas na
memória as variaveis
26 //e redirecionado para a pagina index.php
27 unset ($_SESSION['login']);
28 unset ($_SESSION['senha']);
29 header('location:erro.php');
30 }
31 ?>
32 </body>
33 </html>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

site.php

```
1 <!doctype html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Documento sem título</title>
6 </head>
7
8 <body>
9
10 <?php
11 echo "Deu tudo certo em seu site"
12 ?>
13 <br>
14    <a href="index.php">Inicio</a>
15
16 </body>
17 </html>
```

erro.php

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Documento sem título</title>
6 </head>
7
8 <body>
9
10 <?php
11 echo "sua tentativa deu errado, tente novamente!!!"
12 ?>
13 <br>
14 <a href="index.php">Nova Tentativa</a>
15 </body>
16 </html>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

2. Encapsulamento

Os membros de classe que são utilizados internamente, devem estar inacessíveis externamente. É interessante observar que isso não é meramente uma definição, mas sim, um conceito utilizado amplamente, inclusive pela própria natureza.

Há várias formas para entendermos o encapsulamento, porém, talvez a mais importante seja a capacidade de definirmos um novo tipo de informação e fazer com que instâncias desse novo tipo sejam capazes de manter a integridade de suas informações.

A única maneira para mantermos a integridade é verificando os dados que são enviados pelos membros de classe e estabelecendo através destes as regras para a correta atribuição de informações.

Encapsular implica na modificação da visibilidade de atributos de classe e, portanto, foram definidas algumas palavras-chaves para explicitamente definirmos o escopo de cada membro.

carro.php

```
1  <?php
2
3  class Carro {
4  /*
5  public: significa que você tem acesso aos atributos ou metodos
6  tanto na classe quanto nas suas instancias;
7  private: significa que você tem acesso somente dentro da classe
8  */
9
10 public $marca;
11 public $modelo;
12 public $motor;
13
14
15 function getMarca(){
16 return $this->marca;
17 }
18
19 function getModelo(){
20 return $this->modelo;
21 }
22
23 function getMotor(){
24 return $this->motor;
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

```
25 }
26
27 function setMarca($marca) {
28 $this->marca = $marca;
29 }
30
31 function setModelo ($modelo) {
32 $this->modelo = $modelo;
33 }
34
35 function setMotor($motor) {
36 $this->motor = $motor;
37 }
38
39 public function potencia_motor($motor) {
40 if ($motor >= 1.8) {
41 echo "Potencia do motor: Carro potente";
42 }
43 else {
44 echo "Potencia do motor: Carro popular";
45 }
46 }
47 /*
48 Veja que para escrever os atributos com o "echo"
49 precisamos colocar entre chaves {}
50 */
51 public function listaCarro() {
52 echo "
53 <p>Marca: {$this->marca}</p>
54 <p>Modelo: {$this->modelo}</p>
55 <p>Motor: {$this->motor}</p>
56 {$this->potencia_motor($this->motor)}";
57 }
58 }
59 ?>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

index.php

```
1 <html>
2 <body>
3 <h2>Cadastro de carro</h2>
4 <form method="post">
5 <p>Marca:<input type="text" name="marca"/></p><br>
6 <p>Modelo:<input type="text" name="modelo"/></p><br>
7 <p>Motor:<input type="text" name="motor"/></p><br>
8 <p><input type="submit" name="btnCadastro"/>
9 </form>
10
11 <?php
12
13 /* Aqui, acionamos o arquivo que contém a classe */
14 require_once 'carro.php';
15
16 /* verifica se foi clicado no botão para enviar o formulário com os
dados */
17 if (isset($_POST['btnCadastro'])) {
18
19 /* aqui estamos criando um objeto, ou seja, instanciando a classe
carro */
20 $carro = new Carro();
21
22 $carro->marca=$_POST['marca'];
23 $carro->modelo=$_POST['modelo'];
24 $carro->motor=$_POST['motor'];
25 /* aqui listaremos o que foi capturado pelo POST e gravado nas
variaveis da
classe Carro */
26 $carro->listaCarro();
27 }
28
29 ?>
30 </body>
31 </html>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

3. Método construtor e destrutor

Vamos saber o que é um método construtor. Tal método é invocado no momento em que a classe é chamada com o operador **new**. Em outras palavras, ele é chamado no momento que a classe é construída. Por isso tem esse nome sugestivo: **método construtor**.

Considerações:

- Os métodos construtores precisam ser públicos, pois serão acessados de fora a classe.
- Um dos métodos deve ser usado, nunca os dois juntos.
- Esta forma de construção de classe, utilizando o método, é mais eficaz, contribuindo para um maior aproveitamento de código.
- Lembre-se que ao instanciar a classe com o operador new, passe os parâmetros necessários ao construtor.
- Por fim, no método construtor pode ser chamado outros métodos da classe, ou seja, se você quiser e houver a necessidade

index.php

```
1 <html>
2 <body>
3
4 <h2>Cadastro de pessoa</h2>
5
6 <form method="post">
7
8 <p>Nome: <input type="text" name="nome"/></p><br>
9 <p>Parte do corpo: <input type="text" name="parte"/></p><br>
10 <p><input type="submit" name="btnCadastro"/>
11 </form>
12
13 <?php
14
15 // aqui adicionamos o arquivo que contém a classe
16 include ("pessoa.php");
17
18 // verifica-se se foi clicado no botão para enviar o formulário
com os dados
19 if (isset($_POST['btnCadastro'])) {
20
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

```
21 // aqui estamos criando um objeto, ou seja, instanciando a classe
pessoa e
22 // passando um parametro para o construtor
23 // nesse caso o nome da pessoa
24 $pessoa = new Pessoa($_POST['nome']);
25
26 // chamamos o método criar_corpo e passamos o valor do campo do
formulário
27 $pessoa->criar_corpo($_POST['parte']);
28
29 // chamamos o método que irá escrever o corpo
30 $pessoa->mostra_corpo();
31
32 // aqui destruimos o objeto para executar o destrutor
33 unset($pessoa);
34 }
35 ?>
36 </body>
37 </html>
```

pessoa.php

```
1 <?php
2 /*
3 * para declarar uma classe deve utilizar a palavra reservada class
e logo depois o
4 nome da classe, veja que no nome da classe
5 * a primeira letra está em maiúsculo para justamente diferenciá-la
6 */
7 class Pessoa {
8 public $nome;
9 private $corpo;
10
11 /** * Método construtor */
12 function __construct($n) {
13 $this->nome = $n;
14 echo "<p>Olá {$this->nome} seja bem vindo(a)</p>";
15 }
16
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

```
17 /** * Método que cria meu corpo.. * */
18 function criar_corpo($parte) {
19 $this->corpo .= " {$parte} ";
20 }
21
22 // método para mostrar o corpo
23 function mostra_corpo() {
24 echo $this->corpo;
25 }
26
27 // método que será executado quando o objeto for destruido
28 function __destruct() {
29 echo ", eu sou um método destrutor";
30 }
31 }
32 ?>
```

4. Herança

A herança representa uma das principais características da **Orientação a Objetos**, até porque, somos capazes de implementar tipos de dados hierarquicamente. Através do conceito de herança, conseguimos implementar classes de uso geral, que possuam características comuns a várias entidades relacionadas.

Essas classes poderão ser estendidas por outras, produzindo assim, classes mais especializadas e, que implementem funcionalidades que as tornam únicas.

Através da herança, poderemos utilizar propriedades e métodos definidos na superclasse. Uma boa maneira de pensarmos neste conceito é sob a perspectiva de obter objetos mais especializados conforme aumente a hierarquia. Devemos tomar cuidado com o conceito de hereditariedade animal, até porque, os filhotes não possuem, necessariamente, as características dos pais. Já, o conceito de herança na Orientação a Objetos define que, todo herdeiro receberá o conjunto de características definidas como público e privado e, terá acesso total as funcionalidades definidas na superclasse. Assim, a única maneira de restringir os herdeiros é definindo membros privados, até porque, do contrário, todo e qualquer herdeiro poderá alterar qualquer informação.

É comum que classes derivadas sejam novamente utilizadas como base para outras. Assim, somos capazes de estender qualquer classe que não tenha o seu construtor definido como privado.

Se tomarmos como exemplo a ideia de frutas, temos que a classe fruta conterá o código que define as propriedades e funções de todas as frutas, enquanto que a classe Maçã, receberá as funções e atributos de todas as frutas, e implementará as propriedades e funções que somente as maçãs possuem. Toda **classe poderá ser herdada** e para isso, não é preciso fazer nada de especial, ou seja, o uso da herança se resume a definição explicita na declaração de uma nova classe que a mesma será uma "continuação" de outra.

Index.php

```
1 <html>
2 <head>
3 <meta charset="UTF-8">
4 <title>Herança</title>
5 </head>
6 <body>
7 <?php
8 class Pessoa
9 /*
10 a classe Pessoa é uma super classe. É uma classe generica,
11 a partir dela podemos criar classes mais especificas.
12 */
13 {
14
15 //Atributos da classe
16 var $nome;
17 var $endereco;
18 var $idade;
19
20 // metodos que imprimem os atributos da classe
21 function ImprimeDados()
22 {
23 echo "Nome: {$this->nome}<br>";
24 echo "Endereço: {$this->endereco}<br>";
25 echo "Idade: {$this->idade}<br>";
26 }
27 }
28 class Funcionario extends Pessoa
29 /*
30 14
31 classe filha, herda atributos e métodos de sua classe pai,
32 no caso a classe pessoa. Veja o comando extends para
33 criar a herança
34 */
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

```
35 {
36
37 //atributos da classe
38 var $salario;
39 var $cargo;
40
41 /* método criado para calcular o salario liquido e
42 mostrar o mesmo*/
43 function ObterSalario()
44 {
45 $this->salario -=$this->salario * 0.1;
46 echo "Salário:{$this->salario}<br>";
47 }
48
49 /*Aqui temos uma sobre escrita (overriding). Veja
50 que modificamos o funcionamento do metodo da classe pai
51 acrescentando a impressão na tela do salario e o cargo.
52 O Operador parent: serve para chamar o metodo da classe pai*/
53 function ImprimeDados()
54 {
55 parent::ImprimeDados();
56 echo "Salário Bruto: {$this->salario}<br>";
57 echo "Cargo: {$this->cargo}<br>";
58 }
59 }
60 ?>
61 <?php
62
63 //criando o objeto $func e passando valores
64 $func = new funcionario();
65 $func->nome = "carlos Eduardo";
66 $func->endereco = "Assis Brasil, 123";
67 $func->idade = 23;
68 $func->salario = 2000;
69 $func->cargo = "Diretor";
70 echo "<b>Dados do funcionário</b><br>";
71
72 //passando para o nosso objeto o metodo de mostrar os dados
73 echo "{$func->ImprimeDados()}";
74 echo "<b>Salário do funcionário com desconto de 10%</b><br>";
75
76 //mostrando o salario com o metodo que desconta 10%
77 echo "{$func->ObterSalario()}";
78 ?>
79 </body>
80 </html>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

5. Abstração

Trata-se de criar classes que apenas servirão de modelos para outras classes, sem a possibilidade de serem instanciadas diretamente.

Também existem os métodos abstratos, que servem para definir que um método específico deverá, obrigatoriamente, existir em uma subclasse (classe filha), com ações específicas da necessidade daquela subclasse.

Para criar uma classe abstrata em PHP, simplesmente utilize a palavra abstract antes da criação da mesma. Veja:

abstrato.php

```php
<?php

// Uma pessoa
abstract class Pessoa
{
// Propriedades de identificação da pessoa
public $nome;
public $sobrenome;

// Exibe a identificação
public function exibe_nome () {
echo $this->nome . ' ' . $this->sobrenome . '<br>';
}
}

// Classe filha (subclasse) da classe Pessoa
class Mulheres extends Pessoa
{
// Os métodos já definidos em Pessoa
}
?>
```

index.php

```php
<?php
// Inclui a classe
Include ('abstrato.php');

// Cria o objeto
$mulheres = new Mulheres;

// Configura as propriedades
$mulheres->nome = 'Rose ';
$mulheres->sobrenome = 'Miranda ';

$mulheres->exibe_nome(); // Rose Miranda
?>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

Veja acima que agora não tenho nenhum erro, o código é executado conforme esperado. Isso acontece porque estou criando um objeto (instanciando) a classe filha (a subclasse) da classe "Pessoa". A superclasse "Pessoa" é abstrata e não pode ser instanciada, porém, sua filha "Mulheres" pode.

Dica: Tenha em mente que eu posso configurar superclasses (classes mãe) e subclasses (classes filhas) como abstratas.

6. Acessando banco de dados

O acesso à banco de dados é um dos pontos fortes desta linguagem. O PHP possui acesso nativo a ADABAS, ORACLE, SYBASE, SQL SERVER, DBASE, INFORMIX, mSQL, MySQL, POSTGRESQL, além de suportar ODBC, fazendo com que o PHP possa trabalhar praticamente com todos os bancos de dados existentes.

Neste artigo vamos ver apenas as funções relativas ao banco MySQL, pois esta dupla PHP/MySQL está sendo preferida por uma boa parte dos desenvolvedores, particularmente no ambiente Linux/Apache.

O MySQL é um servidor SQL e portanto, devemos seguir alguns procedimentos e regras para acesso aos seus dados. Se você está acostumado com o Oracle ou SQL Server não terá dificuldades, mas se você usa somente bancos de dados do tipo Access ou DBF, poderá ter dificuldades em entender o mecanismo usado pelo MySQL, mas vou tentar ser o mais didático possível.

A primeira regra é ter um banco de dados cadastrado e um usuário com acesso à este banco de dados. Vale lembrar que o MySQL não é um banco de dados, e sim um servidor de dados. Tenha isto em mente para entender o exemplo.

Digamos que temos um banco de dados Clientes com o usuário admin e senha admin. O primeiro passo é “logar” ao servidor. Para isso usamos o comando mysql_connect e informamos o servidor, login (usuário) e senha. Veja abaixo:

```
$conn = mysqli_connect (“localhost” , “admin”, “admin”, “Clientes”);
```

Este comando abrirá uma conexão com o MySQL da máquina local (localhost), usando o usuário admin cuja senha também é admin ao banco de dados Clientes.
Uma referência a esta conexão será gravada na variável $conn.

Neste ponto já temos uma conexão com o servidor e já criamos um link com o banco de dados. Agora podemos enviar os comandos SQL que desejarmos.

Index.php

```php
<?php

include("banco.php");

?>
```

Banco.php

```php
<?php
$servidor = "localhost";
$usuario = "root";
$senha = "";
$banco_dados = "clientes";

// Cria conexão
$conn = new mysqli($servidor, $usuario, $senha, $banco_dados);

// Check connection
if ($conn->connect_error) {
die("Falha na conexão: " . $conn->connect_error);
}

$consulta = "SELECT id, nome, telefone, email FROM clientes";
$resultado = $conn->query($consulta);

if ($resultado->num_rows > 0) {

// consulta cada linha
while($linha = $resultado->fetch_assoc()) {
echo "ID: " . $linha["id"]. " / CLIENTE: " . $linha["nome"]. " /
TELEFONE: "
. $linha["telefone"]. " / E-MAIL: " . $linha["email"]. "<br>";
}
} else {
echo "0 results";
}
$conn->close();
?>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

Banco de dados:

```
Database:

`clientes`

Estrutura da tabela `clientes`


CREATE TABLE `clientes` (
`id` int(255) NOT NULL,
`nome` varchar(50) NOT NULL,
`telefone` varchar(100) NOT NULL,
`email` varchar(100) NOT NULL
ENGINE=MyISAM DEFAULT CHARSET=latin1;


INSERT INTO `clientes` (`id`, `nome`, `telefone`, `email`) VALUES
(1, 'JUDSON SANTANA', '(61) 34477520', 'contato@versatildf.com.br'),
(2, 'GIOVANNA SANTANA', '(61) 33401815', 'giovanna@versatildf.com.br');
```

Estes comandos devem ser inseridos no banco de dados, ou vá na pasta de exercícios, e restaure o backup do banco de dados pelo PhpMyAdmin.

SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

7. PDO

PDO é uma classe desenvolvida especificamente para trabalhar com procedimentos relacionados a Banco de Dados. O interessante em utilizar este tipo de classe é a abstração de qual banco utilizamos e a segurança extra que esta classe nos oferece.

Quando falamos em "abstração de banco de dados" estamos simplesmente querendo dizer que o PDO nos oferece recurso suficiente para trabalhar implementando toda nossa aplicação sem se preocupar com qual banco estamos utilizando, isso significa que uma mudança posterior na escolha do banco não trará grandes problemas a sua aplicação.

É óbvio que não basta só aplicar o PDO e tudo estará seguro e perfeito. Você precisa também implementar algum padrão de projeto que implemente o conceito de multicamadas, onde o principal objetivo é separar a camada de dados (onde ficam as instruções SQL) do restante da aplicação. E qual a finalidade disso juntamente com o PDO? Separando a sua aplicação em camadas, e utilizando o PDO adequadamente, você terá a receita perfeita de uma aplicação bem implementada e estruturada.

Códigos do banco e index abaixo:

Index.php

```
1 <?php
2 echo "<table style='border: solid 1px black;'>";
3 echo "<tr><th>Id</th><th>Nome</th><th>Telefone</th><th>E-
mail</th></tr>";
4
5 class TableRows extends RecursiveIteratorIterator {
6 function __construct($it) {
7 parent::__construct($it, self::LEAVES_ONLY);
8 }
9
10 function current() {
11 return "<td style='width:150px;border:1px solid black;'>" .
parent::current
(). "</td>";
12 }
13
14 function beginChildren() {
15 echo "<tr>";
16 }
17
18 function endChildren() {
19 echo "</tr>" . "\n";
20 }
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

```
21 }
22
23 $servername = "localhost";
24 $username = "root";
25 $password = "";
26 $dbname = "clientes";
27
28 try {
29 $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
30 $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
31 $stmt = $conn->prepare("SELECT id, nome, telefone, email FROM
clientes");
32 $stmt->execute();
33
34 // define o array resultante para associativo
35 $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
36 foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll()))
as $k=>$v) {
37 echo $v;
38 }
39 }
40 catch(PDOException $e) {
41 echo "Error: " . $e->getMessage();
42 }
43 $conn = null;
44 echo "</table>";
45 ?>
```

Banco de dados:

Database:

`clientes`

Estrutura da tabela `clientes`

```
CREATE TABLE `clientes` (
`id` int(255) NOT NULL,
`nome` varchar(50) NOT NULL,
`telefone` varchar(100) NOT NULL,
`email` varchar(100) NOT NULL
ENGINE=MyISAM DEFAULT CHARSET=latin1;


INSERT INTO `clientes` (`id`, `nome`, `telefone`, `email`) VALUES
(1, 'JUDSON SANTANA', '(61) 34477520', 'contato@versatildf.com.br'),
(2, 'GIOVANNA SANTANA', '(61) 33401815', 'giovanna@versatildf.com.br');
```

8. Sessões

A sessão web é bem parecida com a sessão de um PC, que ao iniciar, temos que colocar usuário e senha, assim seu computador pode saber quem está usando a máquina e guardar os registros com segurança. Em um site ou sistema web, a sessão é importante quando se quer mais segurança na página ou quando se quer ter um controle de usuário. Também alguns programadores se utilizam deste recurso para guardar informações e também pode-se montar um carrinho de compra de um site de vendas, pois assim vão armazenando-se os itens ou produtos e só no final é que os dados são jogados no banco de dados.

Index.php

```
1 <?php
2 // inicia a sessão e grava nas variaveis( cor e animal)
3 session_start();
4 ?>
5 <!DOCTYPE html>
6 <html>
7 <body>
8
9 <?php
10
11 // Seta as variaveis pelas super variaveis $_SESSSION
12 $_SESSION["cor"] = "verde";
13 $_SESSION["animal"] = "gato";
14 echo "A sessão foi iniciada.";
15 ?>
16
17 </body>
18 </html>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

9. MVC

MVC é o acrônimo de Model, View e Controller:

Model é onde fica a parte lógica da aplicação, ou seja, todos os recursos da sua aplicação (consultas ao banco de dados, validações, lógica de disparo de email...), mas ele não sabe quando isso deve ser feito, a camada de model apenas tem o necessário para que tudo aconteça, mas não executa nada de fato. Na verdade o MVC é "mais aplicado" com requisição e tratamento de dados relacionados ao banco, o que acontece é que temos um código mais elaborado surgem muito mais camadas, por exemplo, note que eu inclui a "lógica de disparo de email", mas na verdade isso acaba sendo feito em outras partes da aplicação, por exemplo, no CakePHP temos o Mailer, que é uma camada totalmente independente para se trabalhar com emails, para o Zend Framework tem um pacote para isso e o Laravel usa o SwiftMailer através do recurso Mail, nenhum deles está explicito na camada Model, embora seja o lugar mais obvio (se alguém me forçar a categorizar).

View é onde fica todo o necessário para exibir dados, e isso pode ser muito amplo, imagine um componente para renderizar formulários ou tags HTML, ou mesmo um menu multinível, toda essa lógica fica na view, assim como a Model, a View não sabe quando deve ser executada, ela apenas sabe como fazer, não quando.

Controller é onde tudo acontece realmente, ele é o cara que não sabe como fazer, mas sabe quando. Muitos dizem que o controller não deve ter regras de negócio e automaticamente já entendem que não se deve ter estruturas de controle (if, else...), na verdade é obrigação do controller dizer o que deve acontecer e quando, então imagine que você tem uma página com formulário, como você faria para identificar se a requisição foi pelo formulário o não? Então, o controller deve saber quando fazer as coisas e você deve implementar o necessário para que isso se torne realidade.

Index.php - VIEW

```
1 <html>
2 <body>
3
4 <h2>Cadastro de empregado</h2>
5 <form method="post">
6 <p>Nome: <input type="text" name="nome"/></p>
7 <p>Sobrenome: <input type="text" name="sobrenome"/></p>
8 <p>Salario: <input type="text" name="salario"/></p>
9 <p><input type="submit" name="btnCadastro"/>
10 </form>
11
12 <?php
13 // aqui adicionamos o arquivo que contém a classe
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

```
14 include ("empregado.php");
15
16 // verifica-se se foi clicado no botão para enviar o formulário com os
dados
17 if (isset($_POST['btnCadastro'])) {
18
19 // aqui estamos criando um objeto, ou seja instanciando a classe
empregado
20 $fulano = new Empregado();
21
22 // usando o objeto criado para cadastrar o empregado usando as
informações digitadas no formulário
23 $fulano->cadastro($_POST['nome'], $_POST['sobrenome'], $_POST[
'salario']);
24
25 // por último listamos os dados do empregado conforme está em nossa
classe
26 $fulano->listaEmpregado();
27 }
28 ?>
29
30 </body>
31 </html>
```

Empregado.php – Controler

```
1 <?php
2
3 /*
4 * para declarar uma classe deve utilizar a palavra reservada class e logo
depois o nome da classe, veja que no nome da classe
5 * a primeira letra está em maiusculo para justamente diferencia-la
6 */
7 class Empregado {
8
9 // Aqui cria-se os atributos, ou seja as variaveis que irão armazenar os
dados pertinentes a representar a classe
10 public $nome;
11 public $sobrenome;
12 public $salario_mensal;
13
14 /*
15 * Esse método será utilizado para cadastrar um empregado, veja que está
sendo passado três parametros para o método
16 * irão representar os atributos respectivamente: nome, sobrenome, salario
minimo
17 *
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

```
18 * Os métodos representam as ações que uma classe pode executar
19 * */
20 public function cadastro($n, $sn, $sm) {
21
22 /*
23 * Para poder utilizar atributos e métodos da propria classe utiliza-se a
palavra reservada $this seguido do sinal -> e o nome do atributo, ela é uma
instancia da
24 * classe dentro dela mesmo
25 */
26
27 $this->nome = $n;
28 $this->sobrenome = $sn;
29
30 /*
31 * aqui se faz uma validação do valor do salario minimo antes de passar
para
o atributo, verifica-se se o salario minimo é menor do que zero
32 * sendo um valor negativo atribui-se 0.0 ao atributo caso contrario
atribui-se o valor do salario minimo digitado
33 */
34 if ($sm < 0)
35 $this->salario_mensal = 0;
36 else
37 $this->salario_mensal = $sm;
38
39 }
40
41 /*
42 * Este método irá escrever o cadastro do Empregado, nome, sobrenome e
salario
mensal
43 */
44 public function listaEmpregado() {
45
46 /*
47 * Veja que para escrever os atributos com o echo precisamos colocar entre
chaves
48 */
49 echo "
50 <p>Nome: {$this->nome}</p>
51 <p>Sobrenome: {$this->sobrenome}</p>
52 <p>Salario: {$this->salario_mensal}</p>";
53 }
54 }
55 ?>
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br