**Roteiro 2**

**Objetivos:**

*\* Apresentar a sintaxe básica para utilização de Orientação a Objetos em PHP.*

**Ferramentas necessárias**

*Navegador, NetBeans, Servidor Apache.*

# Orientação a Objetos em PHP

*PHP é uma linguagem de programação concebida de forma estruturada que vem adicionando recursos de programação orientada a objetos ao longo de suas versões.*

*Este roteiro apresenta a sintaxe para utilização de classes e objetos em php. Primeiramente vamos considerar a estrutura de criação de uma classe para representar dados de usuário, em php sua estrutura ficaria assim:*

*<?php*

*class Usuario*

*{*

*//atributos*

*//métodos*

*}*

*?>*

*Importante notar que, mesmo que a classe seja relacionada a usuário devemos* ***evitar a utilização de acentos no nome da classe****, da mesma forma que iremos convencionar que a primeira letra da classe será sempre* ***maiúscula****.*

*Agora apresentamos um exemplo de classe com atributos:*

<?php

*class Usuario*

*{*

***$idUsuario; //atributo***

***$nome; //atributo***

***$login;//atributo***

***$senha;//atributo***

*//métodos*

*}*

?>

*Adicionaremos a classe o método construtor agora, sendo que em php este pode ser representado pelo nome da classe ou a palavra reservada* ***\_\_construct****. Nas nossas aplicações utilizaremos* ***\_\_construct***

<?php

*class Usuario*

*{*

*$idUsuario; //atributo*

*$nome; //atributo*

*$login;//atributo*

*$senha;//atributo*

*//método construtor*

***function \_\_construct()***

***{***

***//comandos de inicialização***

***}***

*//métodos*

*}*

?>

***Encapsulamento***

*É um processo no qual se ocultam as características internas de um objeto aqueles elementos que não têm porque conhece-las. Os modificadores de acesso servem para indicar as permissões que terão outros objetos para acessar seus métodos e atributos.*

*Modificadores de acesso em PHP:*

* *public*
* *private*
* *protected*

*Explicando cada um:*

***public***

*É o modificador de acesso padrão, ou seja, quando não se define o modificador de acesso de um método ou atributo, esse é interpretado pelo PHP como public. É o modificador mais permissivo, pois os métodos ou atributos public podem ser acessados em qualquer parte do programa.*

***private***

*É o modificador de acesso mais restritivo. Um atributo ou método private, só pode ser acessado pelo próprio objeto.*

***protected***

*É um modificador de restrição média. Um atributo ou método protected, pode ser acessado pelo próprio objeto e por objetos das classes filhas (que o herdou).*

*Exemplo de aplicação de encapsulamento em php:*

<?php

*class Usuario*

*{*

***private*** *$idUsuario; //atributo*

***private*** *$nome; //atributo*

***pubic*** *$login;//atributo*

***public*** *$senha;//atributo*

*//método construtor*

***function \_\_construct()***

***{***

***//comandos de inicialização***

***}***

*//métodos*

*}*

?>

*Agora vamos salvar esse arquivo com o nome de* ***Usuario.class.php*** *e instancia-lo em um arquivo de testes.*

***Chamando as classes no php (Instanciando)***

*Agora devemos criar um arquivo para testar a classe. Precisamos referenciar o arquivo que contém o arquivo php, bem como realizar a criação do objeto correspondente a classe. Abaixo segue um exemplo dessa referência:*

<html>

<?php

include\_once 'Usuario.php';

$objUsuario = new Usuario();

$objUsuario->login = "marcelo";

$objUsuario->senha = "1234"

?>

<body>

<div>Meu login é: <?echo $objUsuario->login?></div>

</body>

</html>

***Variável $this***

*Uma pseudo variável $this está disponível quando um método da classe é chamado dentro de um contexto de objeto.*

*Sua implementação normalmente se encontra dentro do escopo de um método e $this é uma referência para o objeto que chama o método ou o atributo.*

***Métodos GET e SET***

*Na orientação a objetos é recomendável que os atributos da classe sejam encapsulados para serem acessados apenas dentro da própria classe e sua manipulação realizada através de métodos específicos chamados* ***Get e Set.***

*Utilizamos o get sempre que precisarmos obter o valor do atributo e o set sempre que desejarmos alterar o seu valor.*

*Observemos o seguinte exemplo com a implementação de get e set:*

class Usuario {

private $idUsuario;

private $nome;

public $login;

public $senha;

function \_\_construct() {

}

// METODOS GETS

public function getLogin() {

return $this->login;

}

public function getSenha() {

return $this->senha;

}

public function getNome() {

return $this->nome;

}

// METODOS SETS

public function setLogin($login) {

$this->login = $login;

}

public function setSenha($senha) {

$this->senha = $senha;

}

public function setNome($nome) {

$this->admin = $nome;

}

}

*No exemplo foram criados métodos get e set para os atributos da classe, deixando fora apenas o idUsuario, porém este também poderia ter um get/set.*

*Agora o exemplo da instância da classe com a chamada aos métodos get/set*

<html>

<?

include\_once 'Usuario.php';

$objUsuario = new Usuario();

$objUsuario->setLogin("marcelo");

$objUsuario->setSenha("1234");

?>

<body>

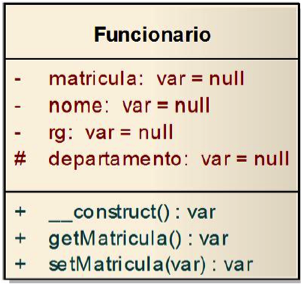
<div>Meu login é: <?echo $objUsuario->getLogin()?></div>

</body>

</html>

***Atividade***

1. *Criar uma classe Funcionario conforme a representação UML abaixo:*

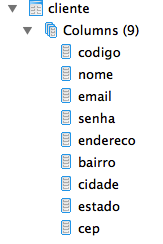
**

*Lembrando que o operador –**indica que o atributo é privado, # atributo protegido e + atributo público.*

1. *Criar um arquivo php que instancie a classe funcionário, atribuindo ao mesmo a matricula* ***123456,*** *exibindo-a no html conforme o texto abaixo:*

*Bem vindo funcionário, seu número de matrícula é* ***123456****.*

1. *Criar uma classe, com os métodos get e set para todos atributos que represente a estrutura da seguinte tabela:*

**

*4. Criar um html que atribua valores a todos os atributos e exiba-os um abaixo do outro, da seguinte forma:*

*Código: 01*

*Nome: João da Silva*

*e assim para todos atributos.*