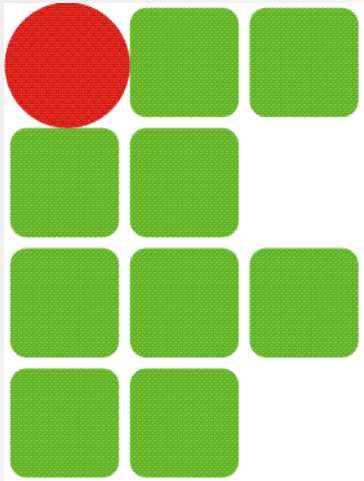


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Campus Bento Gonçalves

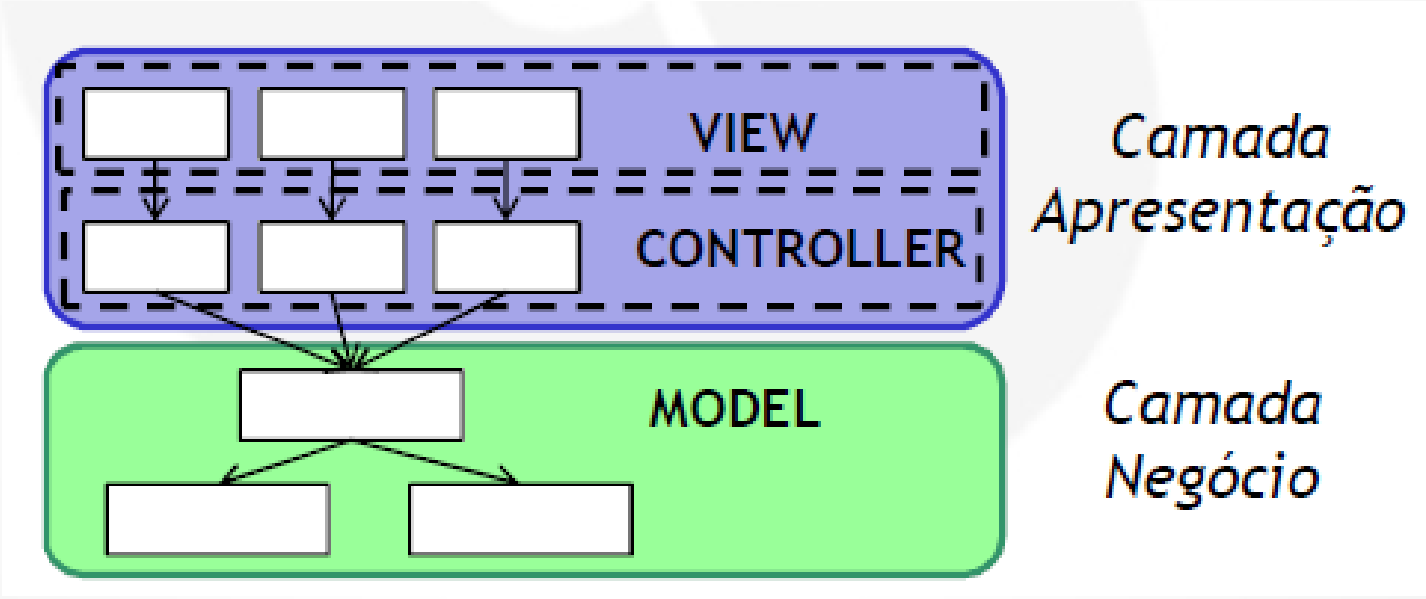
Programação IV

Marcio Bigolin
marcio.bigolin@bento.ifrs.edu.br

MVC

MVC

- O Model-view-controller (MVC) é um padrão de arquitetura de software



ROLLBACK

OO

PHP Orientado a Objetos

- Não tem e provavelmente não terá nenhuma vantagem em relação a outra forma de programação.
- Tudo é uma questão de gosto e prática.
- Mas programando OO você poderá publicar e buscar classes já prontas para fazer alguma função.
- Mas o mesmo aconteceria com a Programação "estruturada".
  - Funções de array
  - Extensões
  - Scripts

Então para que OO

- Reaproveitamento maior de código
- Código de maior legibilidade
- Mais rastreabilidade de seu programa
- Integração de equipes e/ou divisão de tarefas;
- Facilidade na alteração da interface da aplicação;

Reaproveitamento

- Código Estruturado

```
<body>
<?php
$titulo=$_POST["titulo"];
$data=$_POST["data"];
$mensagem=$_POST["mensagem"];
   $link = mysql_connect('localhost', 'user', 'fsfdsfs');
   if (!$link) {
       die('Não foi possível conectar: ');
   }
 mysql_select_db("mural",$link) ;
    $query = "INSERT INTO `mural_diario` (`Titulo`,`dataRecado`,`Mensagem`) VALUES ('$titulo','$data','$mensagem')";
    $result = mysql_query($query);
    if(!$result){
        echo("Sua mensagem não foi postada.");
        exit;
    }
    else{
?>
        <p class="linkSistema">Sua mensagem foi postada no diario com sucesso para o dia :  <?php echo($data); ?> \</P>
        <p> Aguarde enquanto &eacute; redirecionado </p>
        <META HTTP-EQUIV="Refresh" CONTENT="1;URL=../mostrar.php?escrever=0&paraEscrever=20">
<?php
    }
?>
</body>
```

Reaproveitamento

- Código Estruturado

```
<body>
<?php
$titulo=$_POST["titulo"];
$data=$_POST["data"];
$mensagem=$_POST["mensagem"];
   $link = mysql_connect('localhost', 'user', 'fsfdsfs');
   if (!$link) {
       die('Não foi possível conectar: ');
   }
 mysql_select_db("mural",$link) ;
    $query = "INSERT INTO `mural_diario` (`Titulo`,`dataRecado`,`Mensagem`) VALUES ('$titulo','$data','$mensagem')";
    $result = mysql_query($query);
    if(!$result){
        echo("Sua mensagem não foi postada.");
        exit;
    }
    else{
?>
        <p class="linkSistema">Sua mensagem foi postada no diario com sucesso para o dia :  <?php echo($data); ?> \</P>
        <p> Aguarde enquanto &eacute; redirecionado </p>
        <META HTTP-EQUIV="Refresh" CONTENT="1;URL=../mostrar.php?escrever=0&paraEscrever=20">
<?php
    }
?>
</body>
```

Reaproveitamento

- Classe diario

```php
<?php

class Diario{
    private $data;
    private $texto;
    private $titulo;
    function _construct( $titulo,$texto){
        $this->titulo = $titulo;
        $this->data = date("D - m - Y ");
        $this->texto = $texto;
    }
    function setData($data){
        $this->data = $data;
    }
    function SalvarDiario($link, $tabela="diario"){
        $query = "INSERT INTO `$tabela` (`Titulo`,`dataRecado`,`Mensagem`) VALUES
            ('$this->titulo',
             '$this->data',
             '$this->texto')";
        $result = mysql_query($query,$link);
        if(!$result){
            return false;
        }
        return true;
    }


}
?>
```

Legibilidade

- E agora?

```
1  <body>
2  <?php
3  include("diario.php");
4  include("conexao.php");
5  $link = new Conexao('localhost', 'user', 'fsfdsfs', 'mural');
6  $diario= new Diario($_POST["titulo"],$_POST["mensagem"]);
7  $diario->setData($_POST["data"]);
8  if($diario->SalvarDiario($link)){
9  ?>
10     <p class="linkSistema">Sua mensagem foi postada no diario com sucesso
11         para o dia : <?php echo $_POST["data"]; ?></P>
12     <p>Aguarde enquanto &eacute; redirecionado</p>
13     <META HTTP-EQUIV="Refresh" CONTENT="1;URL=../mostrar.php?escrever=0&paraEscrever=20">
14 <?php
15 }
16 else{
17     echo("Erro ao postar mensagem");
18 }
19 ?>
20 </body>
21
```

Integração com a equipe

- Simples
  - pensando no modelo anterior bastaria alguma outra pessoa implementar a classe Conexao()(BD)

Rastreabilidade

- Outra vantagem é que podemos utilizar ferramentas UML
- geram codificação para o PHP
  - livre
    - ArgoUML
    - StarUML
  - Comerciais
    - RationalRose
    - Poseidon

Entendendo MVC

- Classe Veiculo

```php
<?php
class Veiculo {
/**
* Contrutor da classe
*
*/
public function __contruct(){

}
/* qualquer veiculo tem os seus metodos padroes de */
public function andar(){
//.... codigo aqui
}

public function parar(){
//.... codigo aqui
}
}
?>
```

Entendendo MVC

- Classe Automovel herdando métodos de veiculo.

```php
<?php
class Automovel extends Veiculo {

// Contrutor
public function __construct(){

}

public function ligar(){
//.... codigo aqui
}

public function desligar(){
//.... codigo aqui
}

}
?>
```

Entendendo MVC

•Um arquivo qualquer instanciando automovel.

```php
<?php
include("automovel.php");
$meuVeiculo = new Automovel();
$meuVeiculo->ligar();
$meuVeiculo->andar();
$meuVeiculo->parar();
$meuVeiculo->desligar();
?>
```

•Desta forma, se quisermos mudar o comportamento de um Veiculo (independente do que ele seja) basta modificar a classe Veiculo, que todos os tipos de veículos serão alterados (pois eles estendem a classe veiculo) Isso facilita muito na manutenção de código...

Entendendo MVC

•Trabalhos de persistência de dados e comunicação com o banco também fica na camada denominada Model. Porém mesmo assim trabalhamos essa parte de forma separada para não precisarmos nos preocupar com que banco utilizaremos.

```php
<?php
class Model {
public function __construct(){

}
public function conectar(){
// codigo pra conectar ao banco de dados
}
public function query(){
// codigo pra executar uma query
}
public function desconectar(){
// codigo pra desconectar do banco
}
}
?>
```

Entendendo MVC

- Bom vamos a camada view.

```php
<?php
class View {
public function __construct(){

}
public function mostrarNaTela( $template ){
// mostra na tela usando o template escolhido, aqui vem o codigo pra
// ele dar include em algum arquivo html
// modificando os valores que ja estao atribuidos a View e etc...
}
public function atribuirValor( $var, $valor ){
// codigo pra atribuir valores na view
}
}
?>

```

Entendendo MVC

•Bom vamos a camada Controler, esta classe ira utilizar métodos que são genéricos para todos os módulos de sua aplicação. .

```php
<?php
class Controller {

public function __construct(){

}

public function redirect( $url ){
header('Location: ' . $url);
}

public function error( $error ){
// se algum erro acontecer, chamo esse metodo, q vai fazer alguma coisa (mandar um email, gravar log.. )
$this->log( $error );
}

public function log( $mensagem ){
// grava uma menagem de log em algum arquivo qualquer definido aqui
}

}
?>
```

Entendendo MVC

•Sistema de login em MVC classe loginModel. (Regras de negocio – Como não fazer)

```php
<?php
class loginModel extends Model {

public function __construct(){

}

/**
* Um metodo para verficiar a senha do usuario
*
* @param string $usuario
* @param string $senha
* @return bool
*/
public function verificarSenhaUsuario($usuario, $senha){
$this->conectar();
$resultado = $this->query("SELECT * FROM usuarios WHERE
                           usuario = $usuario AND
                           senha = $senha");
$this->desconectar();
return $resultado;
}
}
?>
```

Entendendo MVC

- Classe loginView. (Controla o layout)

```php
<?php
class loginView extends View {
public function __construct(){

}
/**
* Exibe a tela de login
*
*/
public function exibirTelaLogin(){
$this->mostrarNaTela('login.htm');
}
/**
* Exibe a tela de erro
*/
public function exibirTelaErro(){
$this->mostrarNaTela('erro.htm');
}

public function exibirTelaLogado(){
$this->mostrarNaTela('logado.htm');
}

}
?>
```

Entendendo MVC

- Classe loginController ou seja quem “faz o trabalho”.

```php
<?php
class loginController extends Controller {
    // o loginController vai agregar loginModel e loginView
    //entao criamos esses objetos aqui
    private $model;
    private $view;
    public function __construct(){
        // instanciamos os objetos
        $this->model = new loginModel();
        $this->view = new loginView();
    }

    public function telaLogin(){
        // o metodo contrutor ja chama a view do login pra exibir a tela de login
        $this->view->exibirTelaLogin();
    }

    public function fazerLogin(){
        // tenta ver se o usuario colocou a senha certa
        if( $this->model->verificarSenhaUsuario( $_POST['usuario'], $_POST['senha'] )){
            $this->gravaSessao();
            $this->log('usuario logou');
            $this->view->exibirTelaLogado();
        }
        else{
            $this->log('usuario tentou logar mas nao conseguiu');
            $this->view->exibirTelaErro();
        }
    }
    /**
     * Esses metodos eh private, significando que so podem ser chamando dentro da classe
     */
    private function gravaSessao(){
        // grava a sessao usando session ou cookie(Mantem o usuario logado pelas paginas)
    }
    private function apagaSessao(){
        // limpa a sessao ou cookie (Faz logout do usuario)
    }
}
?>
```

Entendendo MVC

- Pronto agora basta instanciar a classe de controle.

```php
<?php
        $objLogin = new LoginController();
        $objLogin->telaLogin();
?>
```