MVC com PHP em Exemplos Práticos

1) Introdução 2
2) Alguma Teoria 3
3) Exemplos Práticos e simples de MVC em PHP5 5
4) Referências 20
5) Dicas úteis 22

1) Introdução

Wikipédia - http://pt.wikipedia.org/wiki/Mvc
Model-view-controller (MVC) é um padrão de arquitetura de software. Com o aumento da complexidade das aplicações desenvolvidas torna-se fundamental a separação entre os dados (Model) e o layout (View). Desta forma, alterações feitas no layout não afetam a manipulação de dados, e estes poderão ser reorganizados sem alterar o layout.

O model-view-controller resolve este problema através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o usuário, introduzindo um componente entre os dois: o Controller. MVC é usado em padrões de projeto de software, mas MVC abrange mais da arquitetura de uma aplicação do que é típico para um padrão de projeto.

Muitos Frameworks podem parecer muito atraentes à primeira vista, porque eles parecem reduzir o desenvolvimento de aplicações web para um par de passos triviais que levam a alguma geração de código e muitas vezes a detecção automática do esquema do banco, mas estes mesmos atalhos são susceptíveis de serem seus gargalos, bem como, uma vez que alcançar esta simplicidade pode sacrificar a flexibilidade e o desempenho.
Rasmus Lerdorf (Criador do PHP).

Portanto não adianta apenas facilitar a criação, mas um framework também deve continuar flexível e permitir uma fácil manutenção do aplicativo gerado. Caso contrário o framework estará dando com uma mão e retirando com a outra.

Aqui é até redundante dizer que para os que estão querendo aprender sobre MVC, a experimentação prática dos exemplos é imprescindível, portanto experimente, altere, personalize e teste bastante até entender e ficar satisfeito.

Em caso de dúvidas, preferencialmente troque idéias com os colegas através do forum do site: efetue seu login e vá em Forum, para que assim as dúvidas e dicas, juntamente com as respostas fiquem à disposição de todos e com isso também estou tentando estimular uma maior integração de todos nós que estamos aprendendo sobre MVC.

2) Alguma Teoria

Abstração de Objetos

PHP 5 introduz métodos e classes abstratos. Não é permitido criar uma instância de uma class que foi definida como abstrata. Qualquer classe que contém pelo menos um método abstrato deve também ser abstrata. Métodos definidos como abstratos simplesmente declaram a assinatura do método, eles não podem definir a implementação.

Quando uma classe herda uma classe abstrata, todos os métodos marcados como abstratos na declaração da classe-pai devem ser definidos na classe filha; além disso, esses métodos devem ser definidos com a mesma (ou menos restrita) visibilidade. Por exemplo, se um método abstrado é definido como protected, a implementação da função deve ser definida ou como protected ou public, mas não private.
http://www.php.net/manual/pt_BR/language.oop5.abstract.php

O PHP tem métodos internos que ajudam quando temos vários campos:

__set() é executado ao se escrever dados para membros inacessíveis.
__get() é utilizados para ler dados de membros inacessíveis.
http://www.php.net/manual/pt_BR/language.oop5.overloading.php#language.oop5.overloading.members

Autoloading Objects

Muitos desenvolvedores ao desenvolver aplicações orientadas a objeto criam um arquivo PHP para cada definição de classe. Um dos maiores contratempos é ter de escrever uma longa lista de includes no início de cada script(um include para cada classe necessária).

Com PHP 5 isso não é mais necessário. Você pode definir uma função __autoload que é automaticamente chamada no caso de você tentar usar uma classe/interface que ainda não foi definida. Ao chamar essa função o 'scripting engine' tem uma última chance para carregar a classe antes que o PHP falhe com erro.
http://www.php.net/manual/pt_BR/language.oop5.autoload.php

Interfaces de Objetos

Interfaces de Objetos permite a criação de código que especifica quais métodos e variáveis uma classe deve implementar, sem ter que definir como esses métodos serão tratados.

Interfaces são definidas usando a palavra-chave 'interface', da mesma maneira que uma classe comum, mas sem nenhum dos métodos ter seu conteúdo definido.

Todos os métodos declarados em uma interface devem ser public, essa é a natureza de uma interface.

implements

Para implementar uma interface, o operador implements é usado. Todos os métodos na interface devem ser implementados na classe; não fazer isso resultará em um erro fatal. Classes podem implementar mais de uma interface se assim for desejado, separando cada interface com uma vírgula.
http://www.php.net/manual/pt_BR/language.oop5.interfaces.php

3) Exemplos Práticos e Simples de MVC com PHP5

A idéia para este curso partiu de um exemplo realmente mínimo que encontrei sobre MVC no PHP5.

Após executar com sucesso comecei a mexer no exemplo, fazendo alterações e verificando o fluso das informações. Percebi que o fluxo não anda no sentido da sigla MVC, nesse sentido, pois geralmente não se admite a comunicação entre as camadas M (model) e V (view). No caso as informações andam nesse sentido:

Fluxo das Informações no MVC

- Geralmente Nascem na View quando um usuário faz uma solicitação, clicando num botão submit ou num link
- Daí são enviadas para o Controller, que a filtra (se for o caso) e a envia para o Model
- O Model analisa de acordo com a solicitação (uma consulta ao banco) e a devolve ao Controller
- O Controler por sua vez devolve o resultado para a View
- E a View renderiza o resultado e o mostra para o usuário

Dizer o que faz cada uma das camadas praticamente todas as definições dizem, mas algo que é de muita importância para o programador, que é o fluxo das informações, onde elas começam e onde terminam, isso já não é comum.

Primeiro eu executei o exemplo e funcionou bem. Depois então, com mi nhas alterações eu saí rastreando as informações para, na prática, perceber como caminhavam.

O exemplo mínimo a que me referi encontrei aqui:

http://www.sourcecode4you.com/article/articleview/5dfe0fd3-5808-4fb2-818e-51c807cf8c6a/mvc-architecture-in-php.aspx

Dando uma olhada vi que usa o banco test e com apenas uma pequena tabela:

```
create table employee
(
    deptid int auto_increment primary key,
    emp_name char(45) not null
);
```

Fiz uma alteração: mudei o nome do banco para mvc.

Inserir alguns rgistros:

```
insert into employee values (default, 'Ribamar');
insert into employee values (default,'Pedro');
insert into employee values (default,'Tiago');
insert into employee values (default,'João');
insert into employee values (default,'Elias');
```

Também dividi em dois arquivos como sugere o artigo: Employee.php e View.php

Employer.php

```
<?php
//////////////My Property Class////////////
// it contain getter setter function

abstract class EmployeeProperty
{
        private $deptid;

        public function getDeptId() //Getter
        {
            return $this->deptid;
        }

        public function setDeptId($id) //setter
        {
                $this->deptid=$id;
        }
}

//////////My Interface /////////
interface iEmployee
{
        function getEmployeeName(EmployeeProperty $objEmployeeProperty);
}

/////////My Data access layer//////
// it fetch data from database server
//it model part of mvc

class DALEmployee implements iEmployee
{
    public function getEmployeeName(EmployeeProperty $objEmployeeProperty)
   {
        $con = mysql_connect("localhost","root","ribafs"); //open connection

        if (!$con){
                die('Could not connect to mysql ' . mysql_error()); // error message
        } else {
            mysql_select_db("mvc",$con); // select database
            $result= mysql_query("select emp_name from employee where deptid=".
$objEmployeeProperty-> getDeptId()); // fire query
                mysql_close($con); // close connection
        }
        return $result;
```

```
    }
}

//////////My business logic layer////////
// it is controller part of mvc
class BALEmployee extends EmployeeProperty
{
     public function getEmployeeName(EmployeeProperty $objEmployeeProperty)
    {
         $objiEmployee=new DALEmployee();
        return $objiEmployee->getEmployeeName($objEmployeeProperty);
    }
}
?>
```

View.php

```
<!--//////////////////My View Part/////////////////////////////////////-->

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> New Document </TITLE>
 </HEAD>
 <BODY>
  <?php
  include("Employee.php");
  ///////////////My View Part//////////////////

  $objBALEmployee=new BALEmployee();  //Create object of business logic layer
  $objBALEmployee->setDeptId(1);      // Set Property
  $result= $objBALEmployee->getEmployeeName($objBALEmployee); // excess bll function
  while($row = mysql_fetch_row($result)) // fetch result
  {
       echo $row[0]."<br>"; // display result
  }
  ?>
 </BODY>
</HTML>
```

Veja que esta querendo trazer do banco o emp_name de um empregado com um certo deptid. Portanto precisamos, antes de testar, adicionar pelo menosum registro e com deptid = 1.

Agora eu comecei a mexer, primeiro dividi em 3 arquivos e fiz o include apenas do que o respectivo iria precisar.

Na primeira experiência eu separei o arquivo Employee.php em dois: Model.php e Controller.php. Além disso criei um script para a conexão do banco de dados, chamado connection.inc.php.

Vejamos como ficaram os includes, que são importantes para entender o fluxo das imformações. Se eu não soubesse como elas caminham ou não estivesse preocupado com elas eu simplesmente faria o include de todos na View, mas me interessa incluir apenas na hora certa o arquivo certo.

No Model.php fiz o include apenas do connection.inc.php pois ele precisa conectar ao banco.

No Controller.php fiz o include apenas do Model.php

No View.php apenas o include do Controller.php

Assim conseguimos perceber quem se comunica com quem.

Agora já criei um outro banco mais brasileiro/português para sentir que de fato estava alterando. Criei o **banco funcionario**, com a tabela funcionarios assim:

Tabela

```
create table funcionarios
(
    codigo int auto_increment primary key,
    nome char(45) not null
);

insert into funcionarios (nome) values ('João Brito Cunha');
insert into funcionarios (nome) values ('Pedro Barbosa Abreu');
insert into funcionarios (nome) values ('Gilberto Braga');
insert into funcionarios (nome) values ('Cassimiro Abreu');
insert into funcionarios (nome) values ('João dos Anzóis Pereira');
```

Veja como ficou o código:

connection.inc.php

```
<?php

     $con = mysql_connect("localhost","root","ribafs");

   if (!$con){
       die('Could not connect to mysql ' . mysql_error());
   }else{
         mysql_select_db("funcionario",$con);
    }
?>
```

Model.php

```
<?php
// Exemplo prático e mínimo de MVC em PHP
// http://www.sourcecode4you.com/article/articleview/5dfe0fd3-5808-4fb2-818e-
51c807cf8c6a/mvc-architecture-in-php.aspx

include_once('./connection.inc.php');

/**
  * Classe Abstrata - Propriedades para getters e setters
  */

abstract class FuncionarioProperty{
     private $codigo;
     private $nome;

   public function getCodigo(){
        return $this->codigo;
   }

   public function setCodigo($id){
        $this->codigo=$id;
   }

   public function getNome(){
        return $this->nome;
   }

   public function setNome($id){
        $this->nome=$id;
   }
}

/**
  * Interface
  */

interface iFuncionario{
     function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty);
}

/**
  * Modelo - Camada de Acesso a Dados
  * Acessa o banco de dados e efetua as operações necessárias com
  */

class DALFuncionario implements iFuncionario{
     public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty){
```

```
        $result= mysql_query("select codigo, nome from funcionarios where codigo=".
$objFuncionarioProperty-> getCodigo());
          print "<script>alert('Model');</script>";
     return $result;
     }
}
```

Controller.php

```
<?php
/**
  * Controller - Camada de Lógica de Negócios
  * Processa dados, efetua operações diversas
  * Recebe requisições do usuário através da View, efetua processamentos como validação do
usuário e outras
  * e envia para o Model a solicitação de dados
  * O Model devolve os dados ao Controller e o Controller devolve para a View
  */

include_once('./Model.php');

class BALFuncionario extends FuncionarioProperty{
     public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty){
          $objiFuncionario=new DALFuncionario();
          print "<script>alert('Controller');</script>";
          return $objiFuncionario->getFuncionarioNome($objFuncionarioProperty);
     }
}
```

View.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Exemplo Mínimo</TITLE>
 </HEAD>
 <BODY>
 <H2> MVC - Exemplo Mínimo</H2>

  <?php
     /**
       * View - Envia solicitação de dados ao Controller
       * Ao receber os dados do Controller os apresenta formatados para o usuário
       */

  print "<script>alert('View');</script>";
  include("./Controller.php");
```

```
 $objBALFuncionario=new BALFuncionario();
 $objBALFuncionario->setCodigo(1); // Passa o código do funcionário a retornar

 $result= $objBALFuncionario->getFuncionarioNome($objBALFuncionario);
 print "<script>alert('View');</script>";

 while($row = mysql_fetch_row($result)){
      echo $row[0]."-";
      echo $row[1]."<br>";
 }
 ?>
</BODY>
</HTML>
```

Depois adicionei apenas um index.php chamando o View.php para ficar bonitinho e não precisar abrir diretamente o View.php.

Na view fiz uma pequena modificação, além de trazer o nome, trouxe também o código.

Veja que se o código requerido não estiver no banco precisará alterar para um existente para que a view traga o registro.

Então chegou o momento de ir em frente e agora alterar os arquivos para poder fazer algo de útil, podedo alterar, inserir e excluir registros.

Personalizando o Exemplo de MVC

Mo exemplo alterado que chamei de mvc_minimo3, eu implementei as operações básicas de uso do banco de dados, o CRUD (muito simples), sem nenhum acabamento, apenas com a intenção de mostrar como essas operações funcionam dentro de um aplicativo usando o famoso padrão de projeto MVC e ainda por cima de forma bem simples.

Espero ter conseguido deixar o exemplo ainda simples após as alterações.

Aqui criei uma View para Insert, uma para Update, uma para Delete, uma para Selet de um registro e uma para Select de todos os registros. Acredito que o próximo passo seja mostrar tudo em um único script.

Mas veja que cada uma dessas Views prcisa encontrar métodos respectivos no Controller e no Model.

connection.inc.php

```
<?php

      $con = mysql_connect("localhost","root","ribafs");

    if (!$con){
        die('Could not connect to mysql ' . mysql_error());
    }else{
          mysql_select_db("funcionario",$con);
     }
?>
```

Model.php

```
<?php
// Pequeno Exemplo de MVC - Requer PHP5
// http://www.sourcecode4you.com/article/articleview/5dfe0fd3-5808-4fb2-818e-
51c807cf8c6a/mvc-architecture-in-php.aspx

//////////////Classe de Propriedades///////////
// contém os métodos: getter e setter

include_once('./connection.inc.php');

abstract class FuncionarioProperty{
     private $codigo;
     private $nome;

//Código
   public function getCodigo() //Getter
   {
        return $this->codigo;
   }
```

```
   public function setCodigo($id) //Setter
   {
       $this->codigo=$id;
   }

//Nome
   public function getNome() //Getter
   {
       return $this->nome;
   }

   public function setNome($id) //Setter
   {
       $this->nome=$id;
   }

}

//////////Interface /////////
interface iFuncionario{
     function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty);
     function getFuncionario(FuncionarioProperty $objFuncionarioProperty);
     function insertFuncionario(FuncionarioProperty $objFuncionarioProperty);
     function editFuncionario(FuncionarioProperty $objFuncionarioProperty);
     function delFuncionario(FuncionarioProperty $objFuncionarioProperty);
}


///////// Modelo - Camada de Acesso a Dados//////
// Puxa dados do banco de dados

class DALFuncionario implements iFuncionario{
  public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty) {
    $result= mysql_query("select codigo, nome from funcionarios where codigo=".
$objFuncionarioProperty->getCodigo());
    return $result;
  }

  // Comente todo este método abaixo e execute para analisar a mensagem de erro
  public function getFuncionario(FuncionarioProperty $objFuncionarioProperty) {
    $result= mysql_query("select codigo,nome from funcionarios");
    return $result;
  }

  public function insertFuncionario(FuncionarioProperty $objFuncionarioProperty) {
      $sql="insert into funcionarios (codigo, nome) values (".$objFuncionarioProperty-
>getCodigo().",'". $objFuncionarioProperty->getNome()."')";
```

```
    $result= mysql_query($sql);
    if(!$result) {
         die('Erro na inclusão<br>'.mysql_error());
     }else{
         print "Registro inserido com sucesso!";
     }
  }

  public function editFuncionario(FuncionarioProperty $objFuncionarioProperty) {
      $sql="update funcionarios set nome='". $objFuncionarioProperty->getNome(). "' where
codigo=".$objFuncionarioProperty->getCodigo();
//print $sql;exit;
    $result= mysql_query($sql);
    if(!$result) {
         die('Erro na atualização<br>'.mysql_error());
     }else{
         print "Registro atualizado com sucesso!";
     }
  }

  public function delFuncionario(FuncionarioProperty $objFuncionarioProperty) {
      $sql="delete from funcionarios where codigo=".$objFuncionarioProperty->getCodigo().";";
//print $sql;exit;
    $result= mysql_query($sql);
    if(!$result) {
         die('Erro na exclusão<br>'.mysql_error());
     }else{
         print "Registro excluído com sucesso!";
     }
  }

}
```

Controller.php

```
<?php
//////////Controller - Camada de Lógica de Negócios////////

include_once('./Model.php');

class BALFuncionario extends FuncionarioProperty{
  public function getFuncionarioNome(FuncionarioProperty $objFuncionarioProperty){
       $objiFuncionario=new DALFuncionario();
       return $objiFuncionario->getFuncionarioNome($objFuncionarioProperty);
  }

  public function getFuncionario(FuncionarioProperty $objFuncionarioProperty){
       $objiFuncionario=new DALFuncionario();
       return $objiFuncionario->getFuncionario($objFuncionarioProperty);
```

```
  }

  public function insertFuncionario(FuncionarioProperty $objFuncionarioProperty){
      $objiFuncionario=new DALFuncionario();
      return $objiFuncionario->insertFuncionario($objFuncionarioProperty);
  }

  public function editFuncionario(FuncionarioProperty $objFuncionarioProperty){
      $objiFuncionario=new DALFuncionario();
      return $objiFuncionario->editFuncionario($objFuncionarioProperty);
  }

  public function delFuncionario(FuncionarioProperty $objFuncionarioProperty){
      $objiFuncionario=new DALFuncionario();
      return $objiFuncionario->delFuncionario($objFuncionarioProperty);
  }

}
```

ViewDelete.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller/////////////////
  $objBALFuncionario=new BALFuncionario();
  $result= $objBALFuncionario->getFuncionario($objBALFuncionario);

  print "<h1>Excluir registros</h1>";
?>

<br>
<br>
<form name="form" method="post" action="">
Código<input type="text" name="codigo"><br>
<input type="submit" name="enviar" value="Enviar"><br>
</form>

<?php

if(isset($_POST['enviar'])){
```

```
    $objBALFuncionario->setCodigo($_POST['codigo']); // Passa o código do funcionário a
retornar
    $result = $objBALFuncionario->delFuncionario($objBALFuncionario);
}
?>

 </BODY>
</HTML>
```

ViewEdit.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller/////////////////
  $objBALFuncionario=new BALFuncionario();
  $result= $objBALFuncionario->getFuncionario($objBALFuncionario);

  print "<h1>Editar registros</h1>";
  while($row = mysql_fetch_row($result)){
       echo "<a href='View4.php?codigo=$row[0]&nome=$row[1]'>$row[0]</a>-";
       echo $row[1]."<br>";
  }

if(isset($_GET['codigo'])){
    $codigo=$_GET['codigo'];
    $nome=$_GET['nome'];
}
?>

<br>
<br>
<form name="form" method="post" action="">
Código<input type="text" name="codigo" value="<?php print $codigo;?>"><br>
Nome<input type="text" name="nome" value="<?php print $nome;?>"><br>
<input type="submit" name="enviar" value="Enviar"><br>
</form>

<?php

if(isset($_POST['enviar'])){
    $objBALFuncionario->setCodigo($_POST['codigo']); // Passa o código do funcionário a
```

```
retornar
     $objBALFuncionario->setNome($_POST['nome']);
     $result = $objBALFuncionario->editFuncionario($objBALFuncionario);
}
?>

 </BODY>
</HTML>
```

ViewInsert.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller//////////////////
  $objBALFuncionario=new BALFuncionario();
?>
<h2>Inserir Registros</h2>

<form name="form" method="post" action="">
Código<input type="text" name="codigo"><br>
Nome<input type="text" name="nome"><br>
<input type="submit" name="enviar" value="Enviar"><br>
</form>

<?php

if(isset($_POST['enviar'])){
$codigo=$_POST['codigo'];
$nome=$_POST['nome'];

  $objBALFuncionario->setCodigo($codigo); // Passa o código do funcionário a retornar
  $objBALFuncionario->setNome($nome); // Passa o código do funcionário a retornar

  $result= $objBALFuncionario->insertFuncionario($objBALFuncionario);
}
  ?>
 </BODY>
</HTML>
```

ViewSelectOne.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller/////////////////
  $objBALFuncionario=new BALFuncionario();
  $objBALFuncionario->setCodigo(3); // Passa o código do funcionário a retornar

  $result= $objBALFuncionario->getFuncionarioNome($objBALFuncionario);

  print "<h1>Retornando apenas um registro</h1>";
  while($row = mysql_fetch_row($result)){
       echo $row[0]."-";
       echo $row[1]."<br>";
  }
  ?>
 </BODY>
</HTML>
```

ViewSelectAll.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> MVC - Pequeno Exemplo </TITLE>
 </HEAD>
 <BODY>

  <?php
  include("./Controller.php");

  ///////////////View - Apresentação dos dados recebidos do Controller/////////////////
  $objBALFuncionario=new BALFuncionario();
  $result= $objBALFuncionario->getFuncionario($objBALFuncionario);

  print "<h1>Retornando todos os registros</h1>";
  while($row = mysql_fetch_row($result)){
       echo $row[0]."-";
       echo $row[1]."<br>";
  }
```

```
  ?>
 </BODY>
</HTML>
```

index.php (mas que não contempla todas as views)

```
<h1>Exemplo prático e mínimo do uso do padrão MVC com PHP</h1>

<?php
if(!isset($_GET['conteudo'])){
    $conteudo='ViewSelectAll.php';
    print "<a href=\"index.php?conteudo=$conteudo\"><h3>Retornar todos os registros</h3></a>";
}else{
    $conteudo2='ViewSelectOne.php';
    print "<a href=\"index.php?conteudo2=$conteudo2\"><h3>Retornar um registro</h3></a>";
}

print "<a href=\"./View3.php\"><h3>Inserir Registro</h3></a>";

if(isset($conteudo)){
    include_once($conteudo);
}else{
    include_once($conteudo2);
}
?>
```

4) Referências

Livros:

1) PHP 5 Power Programming
Autores - Andi Gutmans, Stig Sæther Bakken, and Derick Rethans
Editora - PRENTICE HALL - http://www.phptr.com
Download -
http://ptgmedia.pearsoncmg.com/images/013147149X/downloads/013147149X_book.pdf

2) Object-Oriented Programming with PHP5
Autor - Hasin Hayder - http://hasin.wordpress.com/
Editora - Packt Publishing - http://www.packtpub.com

3) Open Source Content Management System Book Sampler
(Drupal, Alfresco, Moodle, Wordpress, Joomla 1.5, Plone, e107, ezPublish)
Autor - David Mercer
Editora - Packt Publishing - http://www.packtpub.com

4) Programando com Orientação a Objetos
Capítulo 1 de desmonstração
Autor - Pablo Dall'Oglio
Editora – Novatec – http://www.novatec.com.br

http://book.cakephp.org/pt/compare/10/Understanding-Model-View-Controller

http://anantgarg.com/2009/03/13/write-your-own-php-mvc-framework-part-1/
http://slimphp.sourceforge.net/

http://oreilly.com/php/archive/mvc-intro.html
http://www.onlamp.com/pub/a/php/2006/03/02/mvc_model.html
http://www.onlamp.com/php/2005/11/03/mvc_controller.html
http://www.onlamp.com/php/2006/01/26/mvc_view.html

http://www.terminally-incoherent.com/blog/2008/10/22/writing-a-minimalistic-mvc-framework-in-php/

http://filesocial.eu.s3.amazonaws.com/i8tupv6/7e595a6a5f33bef4785ce41db891de33c2d0d59d/ProgramacaoOOPHP5.pdf – Elton

http://www.henriquebarroso.com/how-to-create-a-simple-mvc-framework-in-php/#more-1

http://www.phpro.org/tutorials/Model-View-Controller-MVC.html
http://www.phpro.org/downloads/mvc-0.0.4.tar.gz

http://www.revistaphp.com.br/print.php?id=50
http://www.phpit.net/article/simple-mvc-php5/1/?pdf=yes

http://robrohan.com/projects/download.php?file=examples/MvCphp_1.0.zip
http://www.onlamp.com/lpt/a/6438
http://www.fragmental.com.br/wiki/index.php?title=MVC_e_Camadas
http://pt.wikipedia.org/wiki/MVC
http://www.joomla.com.br/-artigos-mainmenu-43/234-como-sorganizados-os-arquivos-no-joomla-15.html?tmpl=component&print=1&page=
http://imasters.uol.com.br/artigo/5795/php/mvc_em_php_com_smarty_-_parte_02/imprimir/

Pequeno Framework
http://skinnymvc.com/

5) Dicas Úteis

```
define('DS', DIRECTORY_SEPARATOR);
define('ROOT', dirname(dirname(__FILE__)));

$url = $_GET['url'];

require_once (ROOT . DS . 'library' . DS . 'bootstrap.php');

function setReporting() {
     if (DEVELOPMENT_ENVIRONMENT == true) {
       error_reporting(E_ALL);
       ini_set('display_errors','On');
     } else {
       error_reporting(E_ALL);
       ini_set('display_errors','Off');
       ini_set('log_errors', 'On');
       ini_set('error_log', ROOT.DS.'tmp'.DS.'logs'.DS.'error.log');
     }
}

/** Check for Magic Quotes and remove them **/

function stripSlashesDeep($value) {
       $value = is_array($value) ? array_map('stripSlashesDeep', $value) : stripslashes($value);
       return $value;
}

function removeMagicQuotes() {
if ( get_magic_quotes_gpc() ) {
       $_GET    = stripSlashesDeep($_GET   );
       $_POST   = stripSlashesDeep($_POST  );
       $_COOKIE = stripSlashesDeep($_COOKIE);
}
}

/** Check register globals and remove them **/

function unregisterGlobals() {
   if (ini_get('register_globals')) {
       $array = array('_SESSION', '_POST', '_GET', '_COOKIE', '_REQUEST', '_SERVER', '_ENV',
'_FILES');
       foreach ($array as $value) {
          foreach ($GLOBALS[$value] as $key => $var) {
             if ($var === $GLOBALS[$key]) {
                unset($GLOBALS[$key]);
             }
          }
```

```
        }
    }
}

/** Autoload any classes that are required **/

function __autoload($className) {
        if (file_exists(ROOT . DS . 'library' . DS . strtolower($className) . '.class.php')) {
                require_once(ROOT . DS . 'library' . DS . strtolower($className) . '.class.php');
        } else if (file_exists(ROOT . DS . 'application' . DS . 'controllers' . DS .
strtolower($className) . '.php')) {
                require_once(ROOT . DS . 'application' . DS . 'controllers' . DS . strtolower($className) .
'.php');
        } else if (file_exists(ROOT . DS . 'application' . DS . 'models' . DS . strtolower($className) .
'.php')) {
                require_once(ROOT . DS . 'application' . DS . 'models' . DS . strtolower($className) .
'.php');
        } else {
                /* Error Generation Code Here */
        }
}
```

Até o próximo curso! :)