



arquitetura MVC em PHP

**Como desenvolver páginas Web de
forma organizada**

IN MEDIO VIRTUS

Autor: António Rodrigues
Data: 4 Junho 2010

Este documento está protegido pelas convenções e legislação sobre direitos de autor e de cópia. O autor não autoriza a cópia total ou parcial do documento ou de qualquer conteúdo textual ou gráfico sem prévia autorização. O seu uso está limitado pelas condições abaixo discriminadas.

O presente guia pode ser utilizado como referência no desenvolvimento de páginas Web, quer de carácter privado, quer de carácter comercial. Mas não pode ser copiado, distribuído, comercializado ou utilizado na criação de outros guias.

O autor autoriza a utilização de referências e de excertos em trabalhos de investigação académica. Não estão contudo abrangidos trabalhos de investigação com fins comerciais ou sobre os quais incidam contrapartidas monetárias.

Qualquer referência a este documento deve incluir o nome do seu autor, assim como a fonte de onde foi acedido.

Este documento foi escrito ao abrigo do novo acordo ortográfico.

Introdução

O presente guia tem como finalidade ajudar a desenvolver páginas Web em PHP assentes na arquitetura MVC. Esta arquitetura permite uma separação bem definida entre a camada de apresentação e a camada de dados e assegura uma estrutura de código fonte bem organizada.

MVC é o acrónimo de *Model-View-Controller*, ou Modelo-Vista-Controlador em português.

No passado era comum ver páginas Web cuja implementação concentrava todas as suas funcionalidades num único ficheiro. Esta abordagem resultava numa mistura de código de apresentação (HTML), de processamento (PHP) e de dados (SQL). Como consequência o código não podia ser reutilizado noutras páginas, a sua manutenção tornava-se dispendiosa e a interpretação do processo por outros programadores extremamente difícil.

O amadurecimento da programação orientada a objetos tornou possível a reutilização de código, simplificando a manutenção e interpretação, mas não resolveu o problema da mistura de código das diferentes camadas.

Existem vários paradigmas e arquiteturas para a separação das várias camadas lógicas de uma aplicação. Mas a arquitetura MVC foi a que se mostrou mais adequada ao desenvolvimento de páginas Web, ou pelo menos foi a mais bem acolhida pela comunidade de programadores.

A implementação da arquitetura MVC que é apresentada neste guia é uma versão simplificada com o objetivo de ajudar a entender o seu funcionamento e abrir as portas para o desenvolvimento de páginas Web seguindo este modelo. Existem no mercado várias *frameworks* baseadas nesta arquitetura prontas a utilizar que são geralmente mais complexas, mas obviamente também mais flexíveis.

A arquitetura MVC

A arquitetura MVC permite uma clara separação entre camada de apresentação e a camada de dados. A camada de apresentação consiste nos componentes que são visíveis para o utilizador, como por exemplo as páginas Web que são exibidas no navegador. A camada de dados consiste nos componentes que representam e armazenam a informação exibida nessas páginas. Entre as duas existe uma camada intermédia que representa o processo de negócio, ou de outra forma, que é responsável pelo processamento dos dados.

A arquitetura MVC contempla estas três camadas e é representada pela figura seguinte.

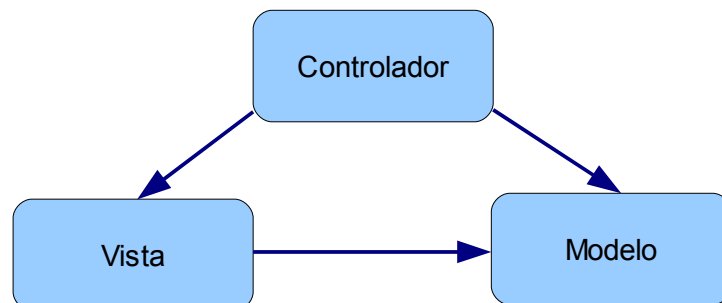


Figura 1: arquitetura MVC

Vista (View)

A Vista (View) representa a camada de apresentação ou a interface com o utilizador. Uma vista contém os componentes visuais e de interação com o utilizador. Numa aplicação Web a vista consiste numa página Web, num formulário de submissão de dados, numa janela de diálogo, ou qualquer outro componente que interage com o utilizador.

A vista conhece o modelo de dados, ou seja, o conteúdo que é disponibilizado na página Web, ou o conteúdo que é pedido num formulário. Por exemplo, para exibir uma tabela com marcas e modelos de automóveis numa página Web é necessário conhecer quais as características dos automóveis disponíveis na camada de dados.

Modelo (Model)

O Modelo (Model) representa a camada de dados, ou mais especificamente, o modelo de dados. O modelo de dados consiste em objetos que reproduzem entidades do mundo real ou tabelas de uma base de dados. Continuando com o exemplo anterior o modelo de dados consistiria numa classe Automóvel com atributos representativos das características de um

automóvel. A lista de automóveis consistiria num conjunto de objetos da classe Automóvel.

O Modelo não conhece absolutamente nada sobre a Vista. O Modelo é uma camada passiva que não possui qualquer funcionalidade além do armazenamento de dados. Desta forma num Modelo nunca se incluem funcionalidades relacionadas com a interface com o utilizador. Por exemplo nenhum dos dados num objeto da classe Automóvel deve conter código HTML, pois este é exclusivo da camada de apresentação e deve ser apenas definido na Vista.

Exemplificando, a classe Automóvel deve conter um atributo com o nome da marca. Mas é da responsabilidade exclusiva da Vista definir qual a cor e tipo de letra usadas para exibir o nome da marca. Tal informação não deve fazer parte do Modelo.

Controlador (Controller)

O Controlador representa o processo de negócio, ou seja é responsável pelo processamento de dados. O Controlador funciona como uma interface entre o Modelo e a Vista.

O Controlador conhece a estrutura de ambos: Vista e Modelo. É ao Controlador que compete a pesquisa de dados para apresentação na Vista, assim como a recolha de dados submetidos pelo utilizador através da Vista (formulários) e o seu armazenamento na camada de dados.

Vejamos dois exemplos. Um sítio Web contém duas páginas Web. Na primeira é exibida uma lista de marcas e modelos de automóveis. Na segunda é permitido a um administrador adicionar mais marcas e modelos de automóveis.

Primeiro caso. A Vista contém o código HTML para exibição de uma tabela onde são apresentadas as características dos automóveis. O Modelo contém os objetos Automóvel, cada qual com atributos representando as características de cada marca e modelo. O Controlador pesquisa uma base de dados de onde recolhe a informação sobre as marcas e modelos de automóveis e preenche os atributos dos objetos Automóvel com os dados pesquisados.

Segundo caso. A Vista contém um formulário que é preenchido e submetido pelo administrador. O Modelo contém um objeto Automóvel que será preenchido com os dados submetidos. O Controlador recolhe os dados submetidos pela Vista, preenche os atributos do objeto Automóvel e armazena esses dados na base de dados.

Implementação em PHP

Neste capítulo é demonstrada a implementação da arquitetura MVC com recurso à linguagem de programação PHP. Os exemplos apresentados são simples, mas é essencial conhecer a linguagem PHP e o paradigma de programação por objetos para uma melhor compreensão.

Reproduz-se aqui o exemplo que tem sido utilizado ao longo deste documento. Imagine-se um sítio Web com uma página que exhibe uma lista de marcas e modelos de automóveis e uma segunda página de administração que permite adicionar novos modelos.

De acordo com a arquitetura MVC o projeto é estruturado de forma a manter uma separação entre os três níveis de abstração. Esta separação é bem visível através dos ficheiros que compõem o projeto e que são apresentados na tabela seguinte.

Vista	Controlador	Modelo
lista.php	ctrlLista.php	Automovel.php
novo.php	ctrlNovo.php	Automovel.php

Tabela 1: Ficheiros PHP em cada camada

A Vista, que representa a camada de apresentação, contém os ficheiros 'lista.php' e 'novo.php'. Estes ficheiros são formados essencialmente por HTML e algum código PHP que exhibe os dados dinâmicos. Como referido anteriormente a Vista conhece o Modelo, logo estes ficheiros incluirão referências aos objetos das classes definidas em 'Automovel.php'.

Os controladores contêm o código essencial ao processo e fazem a ponte entre o Modelo e a camada de apresentação. O controlador 'ctrlLista.php' é responsável por criar a lista de automóveis e disponibilizá-la para a Vista. O controlador 'ctrlNovo.php' é responsável por recolher os dados submetidos através da Vista e inseri-los na base de dados.

O Modelo, que representa a camada de dados, é constituído pelo ficheiro 'Automovel.php'. Por uma questão de simplicidade incluem-se as classes representativas dos dados e de interação com a base de dados no mesmo ficheiro. Num caso real seria preferível manter uma classe por ficheiro de forma a reduzir o tamanho dos ficheiros e consequentemente a quantidade de código interpretada pelo servidor. O nome deste ficheiro é iniciado por maiúscula para indicar que contém classes. Uma alternativa seria utilizar o prefixo 'class' à semelhança de 'ctrl' para o controlador.

É aconselhável agrupar estes ficheiros em diretorias para uma organização mais intuitiva do código. Os ficheiros de apresentação (Vista) devem permanecer na raiz do projeto de forma a serem diretamente acessíveis a partir do navegador. A estrutura final é a seguinte.

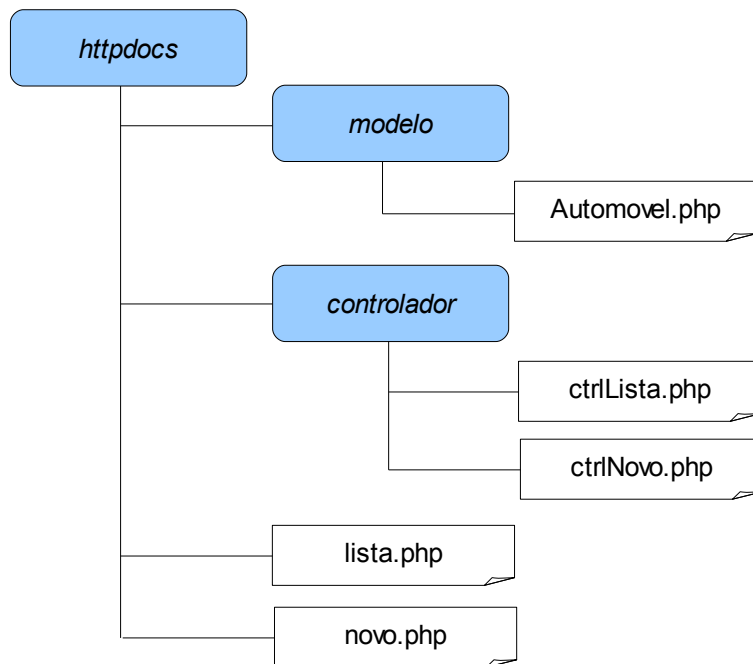


Figura 2: Estrutura de ficheiros e diretorias

Segurança

Os ficheiros controladores e de Modelo devem ser protegidos para que não possam ser acedidos diretamente a partir do navegador. O acesso direto a estes ficheiros força o servidor a executar o seu código PHP tendo resultados imprevisíveis uma vez que os ficheiros de apresentação não são executados.

Existe um simples método que previne a execução do código destes ficheiros no caso de não terem sido acedidos a partir dos ficheiros de apresentação.

No ficheiros de apresentação é definida uma constante de teste. A existência desta constante é verificada nos ficheiros das restantes camadas. Caso um dos ficheiros destas camadas seja acedido diretamente é exibido um erro e a execução do código é interrompida..

Adicionar no inicio dos ficheiros de apresentação 'lista.php' e 'novo.php':

```
define('VALIDO', 1);
```

Adicionar no inicio dos restantes ficheiros:

```
defined('VALIDO') or die('Acesso negado!');
```

Modelo

O Modelo é composto pelas seguintes classes:

- Automovel – Representa os objetos automóvel contendo os atributos destes. Os atributos devem estar em linha com os atributos existentes na base de dados.
- ListaAutomovel – Representa uma lista de objetos automóvel.
- BdAutomovel – Contém as funções necessárias para gerir a base de dados. Inclui funções para a pesquisa de registos de automóvel e a inserção de novos registos.

O exemplo seguinte apresenta o código representativo de cada uma das classes. O código não é totalmente funcional. Fica ao critério do programador implementar as necessárias funcionalidades.

```
class Automovel {
    $marca;
    $modelo;
    $ano;

    _construct() {
        $this->marca = '';
        $this->modelo = '';
        $this->ano = '';
    }
} //fim classe

class ListaAutomovel {
    $lista;

    _construct() {
        $this->lista = new array();
    }
} //fim classe

class DbAutomovel {
    /*
    * Esta função devolve a lista de todos
    * os registos da base de dados
    */
    function getLista() {
        $lista = new ListaAutomovel();

        // conecta com a base de dados
        $conn = mysql_connect(...);
        mysql_select_db(...);

        // copia dados da BD para a lista de objetos Automovel
        $resultSet = mysql_query('SELECT * FROM AUTOMOVEL')
        while ($row = mysql_fetch_assoc($resultSet)) {
            $auto = new Automovel();
            $auto->marca = $row['MARCA'];
            $auto->modelo = $row['MODELO'];
            $auto->ano = $row['ANO'];
            array_push($lista->lista, $auto);
        }
    }
}
```

```

    }

    mysql_close($conn);
    return $lista;
} // fim função

/*
 * Esta função adiciona um novo registo à base de dados
 */
function addRecord($auto) {
    $lista = new ListaAutomovel();

    // conecta com a base de dados
    $conn = mysql_connect(...);
    mysql_select_db(...);

    // insere novo registo
    $sql = "INSERT INTO AUTOMOVEL (MARCA, MODELO, ANO) "
        . "VALUES ('" . $auto->marca . "', '"
        . $auto->modelo . "', '" . $auto->ano . "');"
    $success = mysql_query($sql);

    mysql_close($conn);
    return $success;
} // fim função
} //fim classe

```

Controlador

O Controlador é composto pelos seguintes ficheiros:

- ctrlLista.php – Pesquisa a base de dados e cria uma lista de automóveis que está acessível para utilização na Vista.
- ctrlNovo.php – Obtém os dados submetidos pela Vista com os atributos de uma nova marca e modelo e cria um novo registo na base de dados.

Os exemplos seguintes apresentam o código de cada um dos ficheiros.

Ficheiro ctrlLista.php

```

require_once('modelo/Automovel.php');

$dbAutomovel = new DbAutomovel();
$listAutomovel = $dbAutomovel->getList();

```

Ficheiro ctrlNovo.php

```

require_once('modelo/Automovel.php');

$auto = new Automovel();

// obtém os dados submetidos
if (array_key_exists('marca', $_POST) {
    $auto->marca = $_POST['marca'];
}
if (array_key_exists('modelo', $_POST) {
    $auto->marca = $_POST['modelo'];
}

```

```

}
if (array_key_exists('ano', $_POST) {
    $auto->marca = $_POST['ano'];
}

// insere na base de dados
$success = false;
if (array_key_exists('marca', $_POST) {
    $bdAutomovel = new DbAutomovel();
    $success = $bdAutomovel->addRecord($auto);
}

```

Vista

A Vista é composta pelos seguintes ficheiros:

- lista.php – Exibe a lista de marcas e modelos.
- novo.php – Apresenta um formulário para introdução de dados de um novo registo. Exibe uma mensagem informativa quando o registo é inserido na base de dados.

Os ficheiros são formados essencialmente por HTML que define a apresentação visual da página Web. Os exemplos seguintes apresentam apenas o código relevante para a apresentação de dados ficando ao critério do programador a definição visual das páginas Web.

Ficheiro lista.php

```

<?
    require_once('controlador/ctrlLista.php');
?>

<table class="listAuto">
<? foreach ($listaAutomovel as $key => $item): ?>
    <tr>
        <td class="coluna1"><?= $item->marca ?></td>
        <td class="coluna2"><?= $item->modelo ?></td>
        <td class="coluna3"><?= $item->ano ?></td>
    </tr>
<? endforeach; ?>
</table>

```

Ficheiro novo.php

```

<?
    require_once('controlador/ctrlNovo.php');
?>

<? if ($success) : ?>
    <p>O registo foi criado com sucesso.</p>
<? endif; ?>

<form name="formAuto" class="formAuto" method="post" action="">
    <input name="marca" type="text" class="campoTexto" />
    <input name="modelo" type="text" class="campoTexto" />
    <input name="ano" type="text" size="4" class="campoAno" />
    <input type="submit" value="Submeter" class="botao" />

```

</form>

Conclusão

Como se pode ver pelos exemplos apresentados, a arquitetura MVC permite separar eficientemente e de forma clara as camadas de apresentação, de processamento e de dados.

Esta separação facilita as alterações em cada uma das camadas reduzindo o impacto nas restantes. Também garante uma organização lógica do código que o torna legível e de fácil interpretação. O tempo e esforço investido no desenvolvimento das três camadas acaba por ser recuperado no futuro aquando da necessidade de alterações.

A arquitetura MVC associada a padrões de codificação e de estrutura, vulgarmente conhecidos como '*Code Conventions*' e '*Design Patterns*', permite desenvolver projetos devidamente organizados que poderão crescer com um impacto reduzido no código existente.

Existem no mercado várias '*frameworks*' para PHP que respeitam a arquitetura MVC e que são um bom ponto de partida para projetos mais complexos. A vantagem de algumas destas '*frameworks*' é o facto de incluírem componentes diversos para as três camadas, em resposta às necessidades mais comuns de um sítio Web, que aceleram o desenvolvimento.