O Padrão *Model-View-Controller* Apoiado pelo Framework Zend

Rodrigo Guimarães Bonoto[1], Edson A. Oliveira Junior[2]

Resumo.

Os sistemas de software atuais precisam ser cada vez mais robustos, com uma infinidade de recursos como *feeds*, categorias, formulários, além da necessidade de serem extensíveis e flexíveis. Os padrões de projeto foram concebidos para dar suporte a esses requisitos de sistemas complexos, sendo aplicados para resolverem problemas comuns em projetos. Juntamente com o padrão *model-view-controller* (MVC) permitem melhorar a separação de interesses dos sistemas, diminuindo o esforço com manutenção e a complexidade da arquitetura de tais sistemas. O framework Zend possui uma biblioteca de classes robusta que abrange desde componentes de autenticação, autorização, banco de dados e paginação até componentes que permitem a implementação MVC. Este artigo apresenta uma discussão a respeito do framework Zend, além de um exemplo de aplicação fornecendo suporte ao padrão MVC.

Palavras-chave: *Zend, Framework, Model, View, Controller, MVC*

Abstract. *Current software systems need to be increasingly more robust, with a high number of features such as feeds, categories, forms, beyond the need to be flexible and extensible. Design patterns have been designed to support the requirements of these complex systems, and applied to solve common issues in projects. Along with the standard model-view-controller (MVC) design patterns allow better systems' separation of concerns, reducing the effort for maintenance and architectural complexity of such systems. The Zend framework has a library of classes that range from robust components for authentication, authorization, database and paging controls to MVC implementation components. This paper presents a discussion on the Zend framework, and an application example providing support for MVC.*

Keywords: *Zend Framework, Model, View, Controller, MVC*

1. Introdução

O desenvolvimento de sistemas complexos é uma realidade. Cada vez mais esse tipo de sistema exige formas práticas e efetivas de reduzir o esforço de manutenção e evolução dos sistemas. As tecnologias existentes vêm sofrendo modificações com a necessidade de englobar elementos que permitam diminuir a complexidade inerente dos requisitos e do código-fonte propriamente dito. Para tanto, esforços no desenvolvimento de novos frameworks estão sendo realizados.

Um exemplo do contexto apresentado é a linguagem PHP, difundida por várias razões, como flexibilidade e facilidade de uso. Porém, requisitos mais complexos tornam inviável a adoção simples da linguagem para o desenvolvimento de sistemas de software. Assim, surge a proposta do framework Zend, que segue o padrão *Model-View-Controller*

[1] Aluno do curso de especialização em Desenvolvimento de Sistemas para Web – Universidade Estadual de Maringá (UEM) - Av. Colombo, 5790 – Bloco C56 – Maringá – PR – Brasil – rodrigobonoto@gmail.com

[2] Departamento de Informática – Universidade Estadual de Maringá (UEM) - Av. Colombo, 5790 – Bloco C56 – Maringá – PR – Brasil - edson@din.uem.br

(MVC), visando separar os dados da aplicação (*Model*), da interface do Usuário (*View*) e das regras de negócio da aplicação (Controller) [1].

Este artigo tem como objetivo demonstrar como o framework Zend pode ser usado contribuindo para um bom projeto MVC e proporcionando ganho de desempenho e abreviação no tempo de desenvolvimento de aplicações Web. O modelo estrutural considerado neste artigo leva em consideração a padronização organizacional de aplicações orientadas a objeto com o Zend. O Zend é um framework de código aberto implementado com PHP 5 e licenciado como New BSD Licence [9].

2. Revisão Bibliográfica

2.1 Framework

A utilização de frameworks facilita a organização de uma aplicação, realiza injeção de dependência se necessário, valida campos com lógica de negócio e ajuda a tornar a estrutura do projeto organizada na maioria das vezes dentro de uma modelo MVC [2].

Framework conceitual é usado para resolver um problema de um domínio específico. Esse tipo de framework não se trata de um software executável, mas sim de um modelo de dados para um domínio. Framework de software compreende um conjunto de classes implementadas em uma linguagem de programação específica, usadas para auxiliar o desenvolvimento de software [2].

Um framework atua onde há funcionalidades em comum a várias aplicações, porém para isso as aplicações devem ter um número considerável de elementos em comum para que o mesmo possa ser utilizado em várias aplicações.

Padrões de projeto de software não devem ser confundidos com frameworks, pois padrões possuem um nível maior de abstração. Um framework pode incluir código, diferentemente de um padrão de projeto. Um framework pode ser modelado com vários padrões de projeto, e sempre possui um domínio de uma aplicação particular, algo que não ocorre nos padrões e projeto de software [11].

2.2 O Padrão MVC

MVC faz parte de uma boa prática de projetos de aplicações Web modernas. A maioria do código de aplicações Web está sob uma das três categorias seguintes: apresentação, lógica de negócios e acesso aos dados. Nos modelos do padrão MVC esta separação relaciona-se bem, como ilustra a Figura 1.

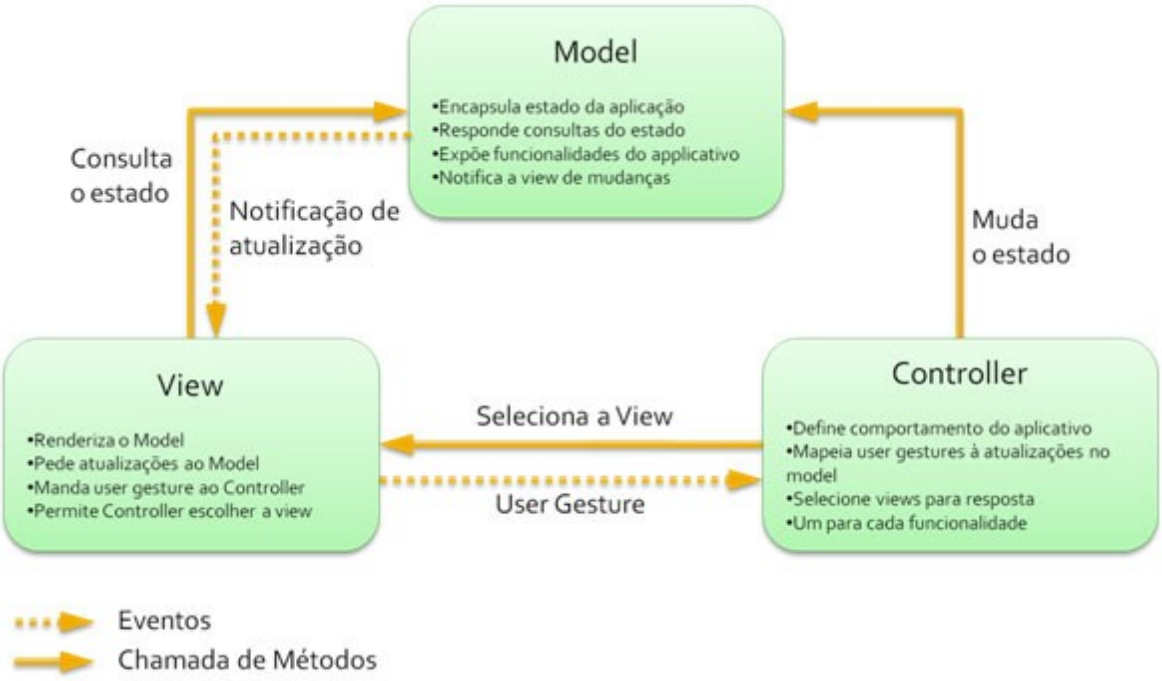


Figura 1: O Padrão *Model-View-Controller* [3].

O resultado final é um código consolidado em uma parte da aplicação com sua lógica de negócio em outra parte e seu código de acesso aos dados também em outra. Muitos desenvolvedores descobriram que essa separação bem definida é indispensável para manter seus códigos organizados, especialmente quando mais do que um desenvolvedor está trabalhando em uma mesma aplicação, trata-se do modelo MVC, acrônimo para *Model-View-Controller.*

Model - É a parte da aplicação que define sua funcionalidade básica por trás de um conjunto de abstrações. Rotinas de acesso a dados e alguma lógica de negócio podem ser definidos no modelo.

View - Apresentações definem exatamente o que será apresentado ao usuário. Geralmente os controladores passam os dados para cada apresentação interpretá-los em algum formato. Apresentações também coletam frequentemente os dados do usuário.

Controller - Controladores vinculam todo o padrão em um conjunto. Eles manipulam modelos, decidem qual apresentação será exibida com base em solicitações do usuário e em outros fatores, repassam os dados que cada apresentação necessita, ou transferem completamente o controle para outro controlador [10].

2.3 O *Framework Zend*

O Zend Framework “trabalha com blocos de construção” que podem ser usados peça por peça com outras aplicações e frameworks, ele também é extensível e fácil para adaptar a estrutura as necessidades, possui várias ferramentas de codificação de criptografia e segurança que você precisa, suporta múltiplos idiomas, e possui uma base de usuários muito ativa para obtenção de ajuda.

O Zend Framework faz parte do projeto PHP Collaboration, uma iniciativa da empresa Zend Technologies. Ele é um framework escrito em PHP5 que utiliza todos os recursos de orientação a objetos fornecidos por esta versão da linguagem, além de prezar

a utilização de padrões de projetos, visando à construção de componentes altamente reutilizáveis [4].

O Zend Framework é flexível, possuí uma coleção de componentes reutilizáveis e extensíveis que podem ser utilizados juntos ou separados, e permite que os desenvolvedores usem qualquer componente da forma que quiser. Além disso, ele oferece abstração de banco de dados em uma forma muito simples de usar, um componente de formulários que implementa e renderiza HTML5, validação e filtragem de modo que os desenvolvedores podem consolidar todas essas operações. Outros componentes, como o Zend Auth e Zend Acl, fornecem autenticação e autorização do usuário em todos os ambientes do sistema. Ele também oferece API's e interface de programação de alguns serviços do Google através do componente Zend_Gdata, que é uma interface em PHP 5 para acessar dados do Google. Este componente suporta o acesso a serviços como [12]:

- Google Calendar – aplicativo de calendário;
- Google Spreadsheets – ferramenta de planilhas que pode ser usado para armazenamento de dados;
- Google Documents List – fornece uma lista de todas as planilhas, documentos, apresentações armazenadas em uma conta do Google;
- Google Provisioning - oferece a capacidade de criar, recuperar, atualizar e excluir contas de usuário, apelidos, grupos e listas de e-mail em domínio do Google;
- Youtube - oferece a possibilidade de pesquisar e recuperar vídeos, comentários, favoritos, assinaturas, perfis de usuários e muito mais;
- Picasa Web Albuns – aplicativo de compartilhamento de fotos;
- Google Analytics – aplicação de estatísticas de visitantes;
- Google Blogger – aplicação para gerenciamento e criação de blogs;
- Google CodeSearch – permite procurar código-fonte de projetos;

O Zend também oferece um robusto e alto desempenho em implementação MVC. Este padrão arquitetural define que uma aplicação será dividida em três camadas lógicas, cada uma com responsabilidades distintas. O *controller* será responsável por conter toda a lógica de domínio de negócio da aplicação, envolvendo desde um cálculo complexo, até a lógica de acesso ao banco de dados. A *view* fica responsável em exibir para o usuário dados do *model*, ou interfaces gráficas para entrada de dados. O Controller é o responsável por intermediar requisições da *view* e delegar lógica do *model*, assim como notificar a *view* sobre possíveis mudanças nos dados do *model* [12].

3. Criando Projetos com o Zend Framework

3.1 Configuração do Ambiente para Desenvolvimento com Zend

Antes de iniciar o desenvolvimento de aplicações com Zend Framework é necessário ter alguma experiência com PHP, e também um ambiente de desenvolvimento com um servidor web de preferência apache funcionando com php. Um banco de dados instalado não é obrigatório, mas será necessário para construir aplicações com acesso a dados.

Tendo um ambiente preparado, já pode ser baixado e instalado individualmente o servidor Apache, PHP, e banco de dados. Além dessa forma existem alguns pacotes prontos com todas essas aplicações para realizar uma única instalação, um desses pacotes é o Zend Server[3], que já inclui o Zend Framework.

Caso o Zend Server tenha sido instalado, o download e instalação do Zend Framework já foi realizado junto com ele, caso contrário deve ser feito o download do Zend Framework baixando preferencialmente o Full Package, dessa forma baixará um arquivo compactado contendo a pasta bin, onde está o zend tool, a pasta library onde estão de fato os arquivos do framework e diversas outras pastas contendo exemplos e bibliotecas de terceiros que não são relevantes neste caso [4].

O Zend Framework permite criar praticamente qualquer estrutura para armazenar os arquivos que compõem uma aplicação. No entanto, isso pode dificultar mais do que facilitar o aprendizado com Zend Framework para o desenvolvedor iniciante. Para tanto, existe uma estrutura recomendada pela Zend, como ilustra a Figura 2.

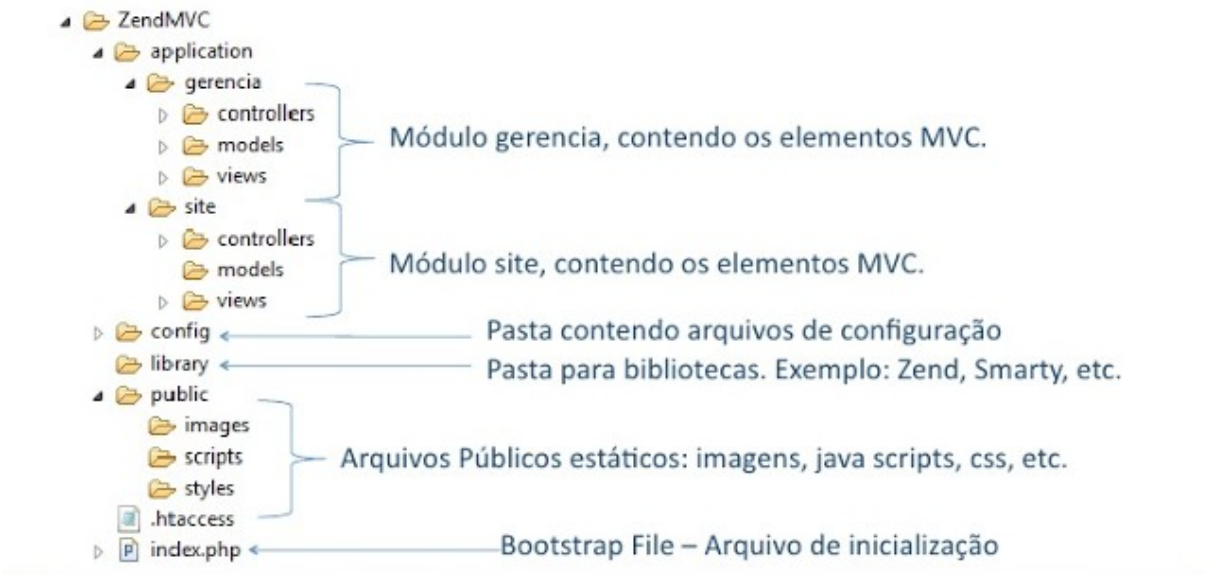


Figura 2: *Estrutura de Diretórios Recomendada para o Zend [5].*

O primeiro modo de criar a estrutura de diretórios do Zend Framework é utilizando o Zend Tool. Caso o ambiente tenha sido configurado com Zend Server, antes de utilizar o Zend Tool os arquivos do Zend Tool devem ser copiados para a pasta de arquivos binários dentro da instalação do PHP. Esses arquivos estão localizados na pasta bin [6].

No caso de ambientes Linux e Unix essa pasta geralmente está localizada em um dos seguintes caminhos:

- /usr/bin
- /usr/local/bin
- /usr/local/ZendServer/bin/
- /Applications/ZendServer/bin/

[3] Zend Server download - http://www.zend.com/en/products/server/downloads

No caso de ambientes Windows, essa pasta geralmente está localizada em um dos seguintes caminhos:

- C:\php
- C:\Program Files\Zend Server\bin
- C:\xampp\php\bin

Caso tenha sido instalado o Zend Server isso já foi feito automaticamente. Agora com o Zend Tool instalado no sistema e adicionado como uma variável path, tudo que é necessário fazer para criar a estrutura de diretórios para o Zend Framework é abrir um console ou *prompt* de comando e navegar até a pasta onde deseja criar a estrutura de diretórios. Isso pode variar entre Windows e ambientes Unix e pode exigir um pouco de conhecimento em linha de comando.

3.2 O Zend Studio

O Zend Studio é um IDE de desenvolvimento baseado em Eclipse e desenvolvido pela própria Zend. Das opções de configuração e criação de projeto essa é a ferramenta que torna este processo mais fácil para quem não tem nenhum conhecimento em linha de comando. Para criar novos projetos é a mais adequada no desenvolvimento de aplicações baseadas em Zend Framework. Além disso, o uso do Zend Studio torna a manipulação e desenvolvimento do projeto diretamente ligado ao Zend Framework, tendo em vista que ele possui integrado a sua instalação as bibliotecas do framework e do PHP, controle de svn, e opções para criação de praticamente todos os objetos usados no desenvolvimento de aplicações baseadas em PHP e Zend [7].

A Figura 3 apresenta a interface da tela inicial do Zend Studio.

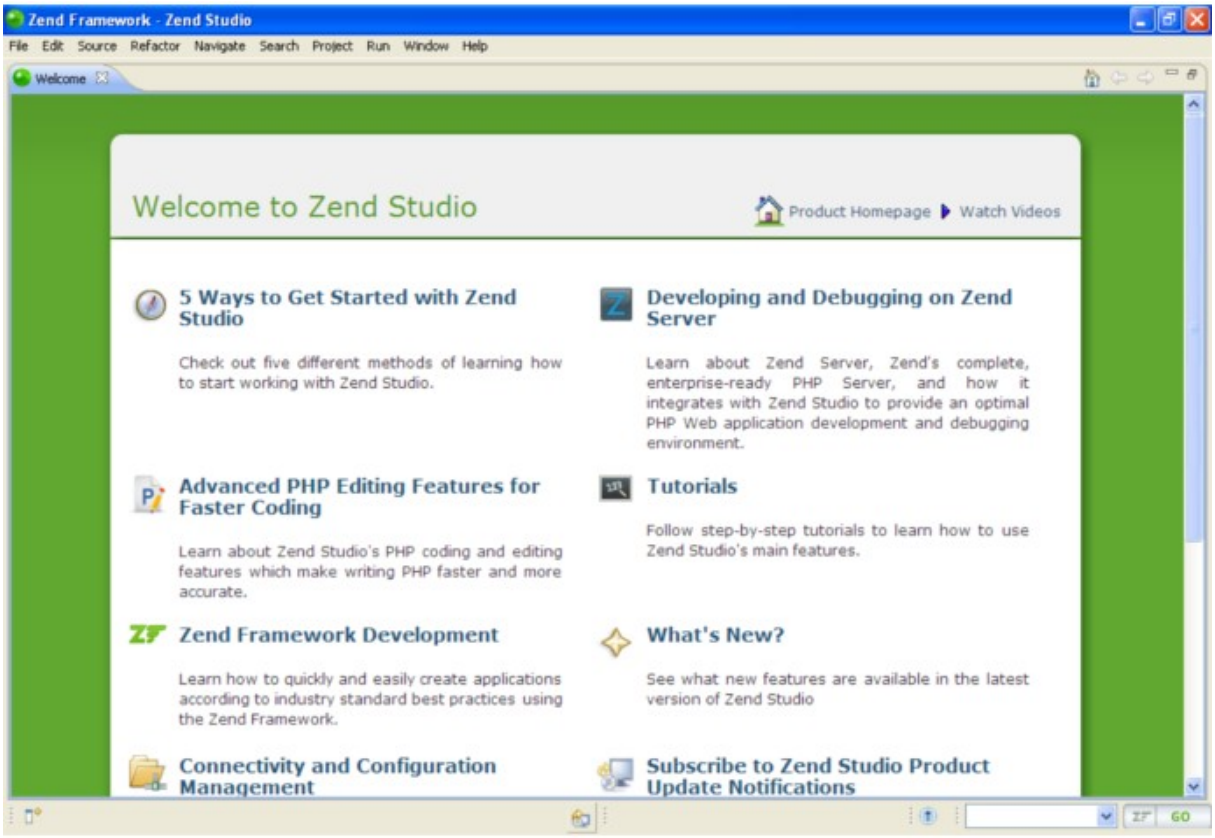


Figura 3: *Interface Zend Studio [8].*

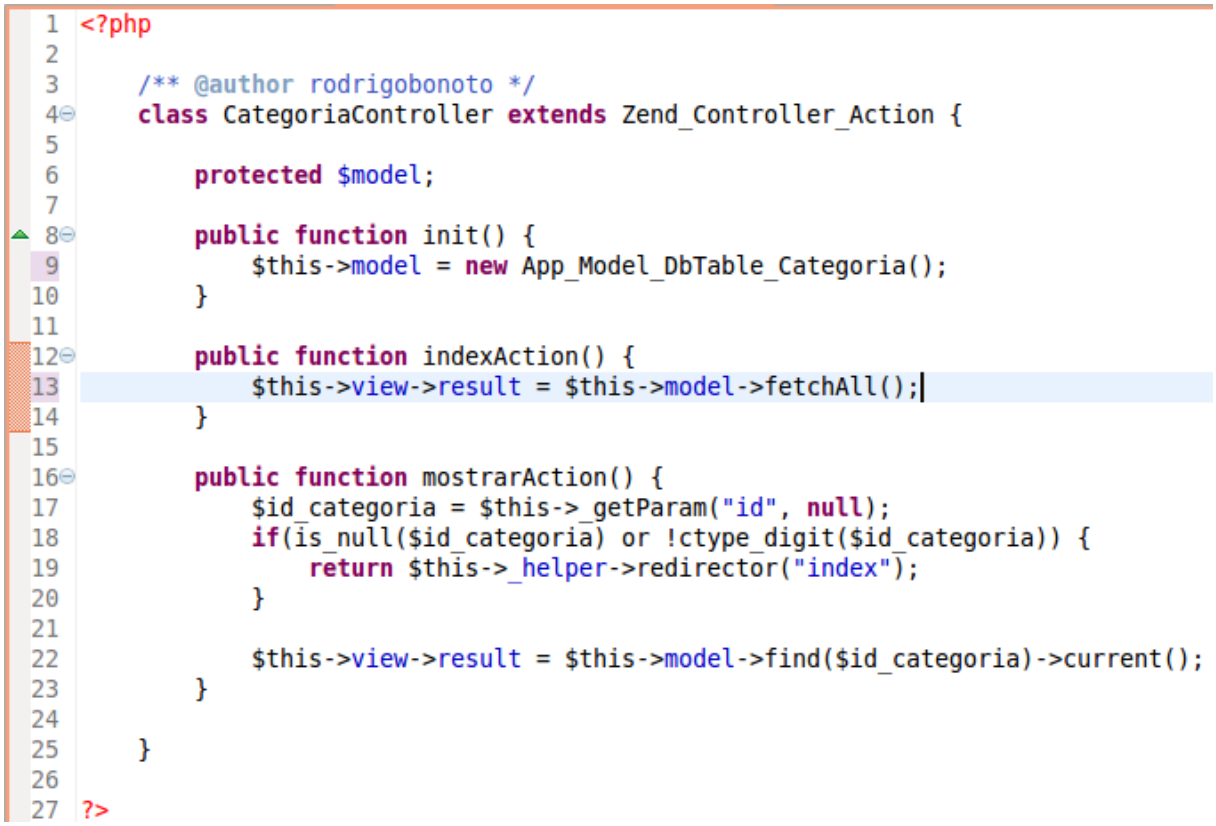
A Figura 3 mostra a IDE para desenvolvimento com Zend Framework. Importante ressaltar que o uso do Zend Studio é imprescindível para o desnevolvimento com o Zend Framework, já que com sua instalação toda a biblioteca PHP e Zend Framework vem "embutida".

3.3 Exemplo de Uso do Zend

Zend_Controller_Action é uma classe abstrata usada para implementar controladores de ação para uso com o *Front Controller* ao construir um website com base no padrão MVC.

A operação mais básica é criar métodos de ação que correspondem às várias ações que se deseja que o controlador execute. A ação executada é identificada através da URI do navegador, onde o primeiro parâmetro corresponde ao controlador (quando o módulo é *default*), o segundo à ação e o terceiro em diante (se houver) são os parâmetros.

Por exemplo, digamos que uma subclasse é definida como na Figura 4.



```php
<?php

    /** @author rodrigobonoto */
    class CategoriaController extends Zend_Controller_Action {

        protected $model;

        public function init() {
            $this->model = new App_Model_DbTable_Categoria();
        }

        public function indexAction() {
            $this->view->result = $this->model->fetchAll();
        }

        public function mostrarAction() {
            $id_categoria = $this->_getParam("id", null);
            if(is_null($id_categoria) or !ctype_digit($id_categoria)) {
                return $this->_helper->redirector("index");
            }

            $this->view->result = $this->model->find($id_categoria)->current();
        }

    }

?>
```

Figura 4: *Controlador de Ações Zend.*

A classe CategoriaController (Figura 4) define duas ações apenas, *index* e *mostrar*. O método init() é executado por último, embutido no construtor padrão da classe Zend_Controller_Action (*__construct*), porém é possível instanciá-lo para personalizar alguma tarefa. O construtor é executado sempre antes de qualquer ação posterior, no caso acima, antes de *indexAction* e *mostrarAction*. Neste exemplo foi usado o init() para enviar a variável *model* a instância da entidade do banco de dados que será manipulada em todas as ações do controlador.

A ação *indexAction* envia para a *view index* todos os registros da entidade categoria por meio do método *fetchAll*. Este método trata-se de uma instrução SQL – Select * From tabela. O retorno dessa execução é “embutido” na variável *result* sendo enviada para a *view* por meio da instrução $this->view->result;

Outras ações podem ser realizadas, como, por exemplo, ações personalizadas de inicialização e uma variedade de métodos auxiliares. Podemos, por exemplo, em vez de usar a instância da tabela do banco de dados, criar um *Model* com instruções SQL pré-definidas, dessa forma a variável $this->model deixaria de receber o DbTable da tabela e passaria a usar o *Model* criado. Assim, as ações deixariam de usar diretamente (se necessário) o método *fetchAll* para listar todos os registros e *find()* para mostrar o registro da chave primária requisitada e passariam a usar os métodos definidos nesta nova classe.

Podemos definir o *model* personalizado como na Figura 5.

```php
<?php

    /** @author rodrigobonoto */
    class App_Model_Categorias extends App_Model_DbTable_Categoria {

        public function getCategoria($id_categoria) {
            $result = $this->find($id_categoria)->current();
            return ($result)? $result : false;
        }

        public function getCategorias($sql = null, $pager = false) {
            $sql = (!$sql)? $this->select()->order(array("nm_categoria")) : $sql;

            if($pager) {
                $result = Zend_Paginator::factory($sql);
                $result->setItemCountPerPage(25);
                $result->setCurrentPageNumber(
                    Zend_Controller_Front::getInstance()
                    ->getRequest()
                    ->getParam("pagina", 1)
                );

                return $result;
            }

            return $this->fetchAll($sql);
        }

    }
?>
```

Figura 5: *Modelo Responsável pela Lógica.*

O *model DbTable* refere-se à entidade no banco de dados. Além disso, é possível definir nela referências de chaves estrangeiras, dependências de outras entidades, chave primária única ou composta, *schema*, entre outras configurações. O DbTable é definido basicamente no formato mostrado na Figura 6.

```php
<?php

    /** @author rodrigobonoto */
    class App_Model_DbTable_Categoria extends Zend_Db_Table_Abstract {

        protected $_name = "categoria";
        protected $_rowClass = "App_Model_Categoria";

        protected $_referenceMap = array(
            "fk_id_midia" => array(
                "columns" => "id_midia",
                "refTableClass" => "App_Model_DbTable_Midia",
                "refColumns" => "id_midia"
            )
        );

    }

?>
```

Figura 6: *DbTable: Definições de Entidade do Banco de Dados e suas Relações.*

As definições da entidade do banco de dados como é mostrada na Figura 6 acima, pode ser definida da seguinte maneira:

- linha 6: Nome da entidade no banco de dado;
- linha 7: Modelagem da estrutura da entidade por meio dos métodos get e set. É nesta classe que será enviada toda requisição de alterações de registro (insert, update e delete).
- linha 9 à 15: Mostra que a tabela categoria possui uma chave estrangeira da entidade mídia, portanto cria-se esta referência para usá-la quando necessária.

A Figura 7 representa a abstração citada na linha 7 da Figura 6:

```php
<?php

    /** @author rodrigobonoto */
    class App_Model_Categoria extends Vivaweb_Model_ObjectBase {

        public function _insert() {
            parent::_insert();
        }

        public function _update() {
            parent::_update();
        }

        public function getIdCategoria() {
            return $this->id_categoria;
        }

        public function getNmCategoria() {
            return $this->nm_categoria;
        }

        public function setIdCategoria($id_categoria) {
            $this->id_categoria = $id_categoria;
        }

        public function setNmCategoria($nm_categoria) {
            $this->nm_categoria = $nm_categoria;
        }

    }

?>
```

Figura 7: *Modelagem da tabela e seus métodos get's e set's*

Na Figura 7 podemos visualizar toda abstração da classe referenciada na linha 7 da Figura 6. Nela serão manipuladas todas as requisições enviadas a entidade.

Basicamente até aqui foram mostradas duas camadas do modelo MVC trabalhado em Zend. Resumidamente, percebemos que o *controller* é o responsável por receber as requisições do navegador. Em cada ação são executados os métodos necessários dentro da camada responsável pela lógica (*model*) e devolve-se o resultado enviando-o para a *view*, área de visualização do que foi requisitado, que são basicamente as páginas do navegador Web.

Portanto, o último passo é mostrar no navegador o resultado das requisições das ações *index* e *mostrar* o controlador CategoriaController.

Como mencionado anteriormente, a ação *indexAction* envia para a *view* todos os registros da entidade categoria, neste caso recebemos a variável "*result*" em nossa *view* e usamos da forma que é conveniente, como mostra a Figura 8.

```
1     <h2>Categorias</h2>
2     <ul>
3 <?php
4     foreach($this->result as $categoria) { /* @var $categoria App_Model_Cegoria */
5 ?>
6         <li><?=$categoria->getNmCategoria()?></li>
7 <?php
8     }
9 ?>
10    </ul>
```

Figura 8: *View: Criando um Laço com o Resultado Enviado.*

A saída (resultado) do navegador ficará da seguinte forma (Figura 9):

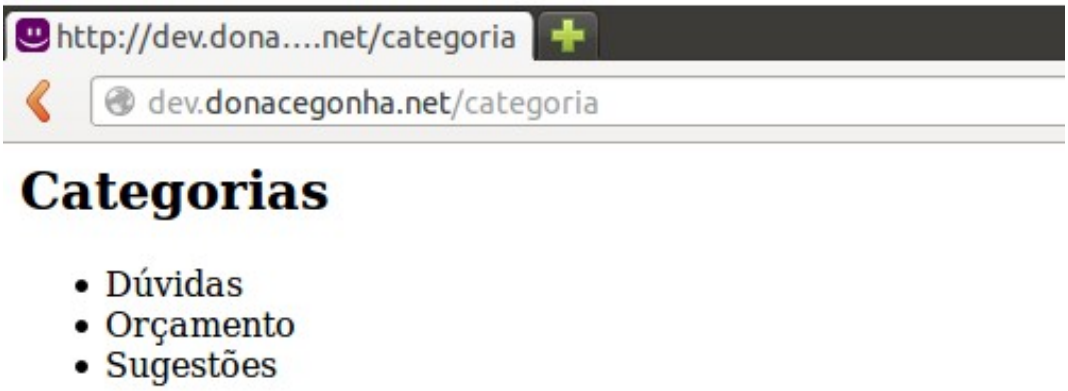


Figura 9: *Saída (Resultado) do Navegador.*

A Figura 10 mostra a saída do navegador quando é passado um parâmetro de uma categoria específica.

Figura 10: *Saída (Resultado) do Navegador para a Categoria Específica.*

Dessa forma, as três camadas do modelo MVC são suportadas pelo Zend Framework: sendo elas:

- C – Camada de controle definida no *controller* CategoriaController;
- M – Camada responsável pela lógica de negócio; e
- V – Camada de visualização da requisição solicitada.

O fluxo MVC para um aplicação baseada em Zend Framework é definido conforme a Figura 11 abaixo:

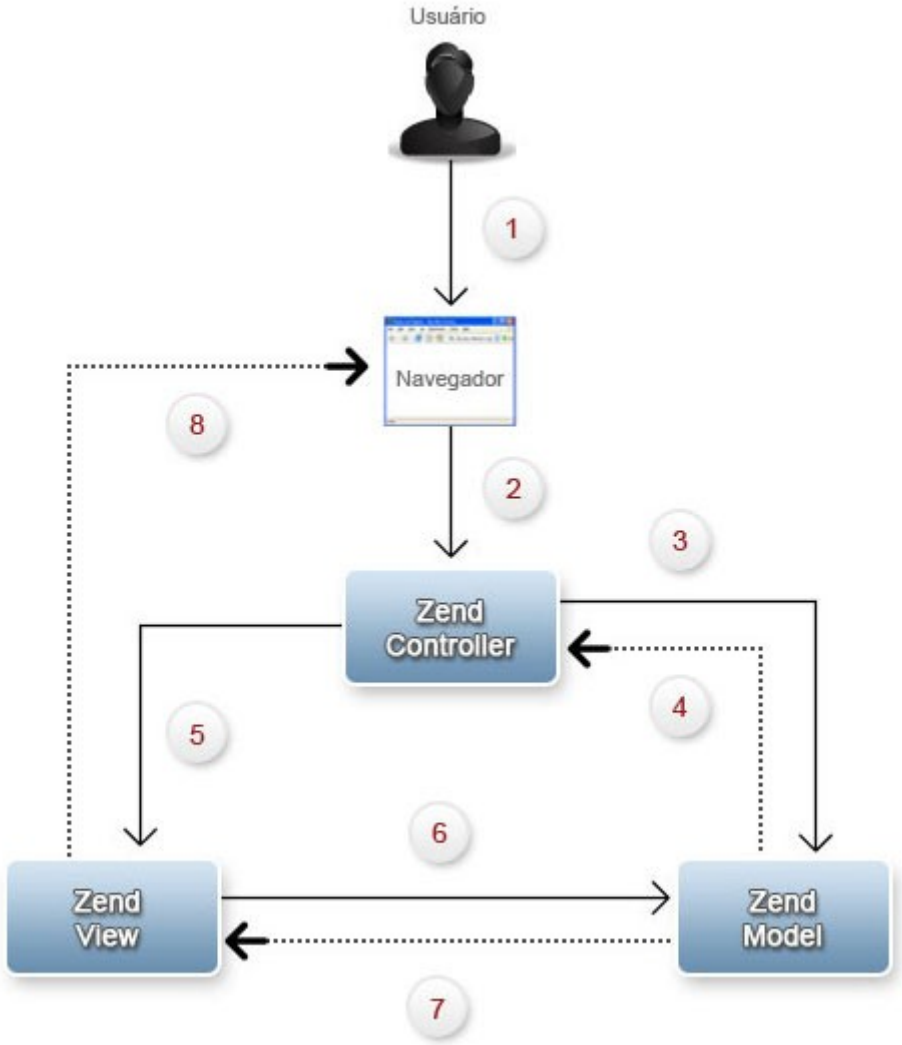


Figura 11: *Fluxo MVC para aplicações em Zend Framework*

1. O usuário abre o navegador e acessa uma aplicação desenvolvida em Zend Framework;
2. O controlador (Zend Controller) responsável pela endereço acessado pelo usuário é acionado, e recebe as requisições que lhe foram passadas, e as delega para a camada responsável pela lógica de negócios da aplicação;
3. Neste momento a camada responsável pela lógica de negócio (Model) é “chamada” e executada através de um componente Zend Model;
4. O “Zend Model” devolve para o Zend Controller os resultados encontrados;
5. O Zend Controller por sua vez envia para o Zend View as informações encontradas;
6. O Zend View recebe essas informações e através de um objeto recupera todo os dados que lhe foram transmitidos pelo Zend Controller no “Zend Model”;
7. O “Zend Model” envia valor por valor para o Zend View de cada objeto solicitado;
8. O Zend View devolve para o navegador através de uma página html e com linguagem de template baseada em PHP as informações solicitadas, e então “encerra-se” o ciclo MVC em uma aplicação Zend para a requisição atual.

4. Considerações Finais e Trabalhos Futuros

Este artigo apresentou a criação de um exemplo utilizando MVC apoiado pelo Zend Framework, com a finalidade de mostrar a viabilidade e a facilidade do uso do padrão MVC com Zend e PHP. A estrutura e as classes apresentadas neste artigo mostra o quanto é organizado, limpo e ao mesmo o quão dinâmico é um projeto em Zend, mostrando que o PHP deixou à muito tempo de ser apenas uma linguagem de programação procedural e consequentemente maçante.

O modelo apresentado neste artigo foi desenvolvido com o objetivo de mostrar a estrutura MVC baseada em Zend Framework, mostrando que o uso deste framework aumenta a performance e economiza tempo no desenvolvimento de aplicações web desenvolvidas em PHP comparado ao desenvolvimento de aplicações com código "artesanal", ou programação procedural em PHP.

Ao contrário de outros frameworks PHP, como por exemplo o CakePHP, o Zend Framework pode separar a aplicação por módulos, o que torna a aplicação mais flexível e extensível e a manutenção mais simples e menos propícia a erros.

Por trabalhar com reescrita de url (mod_rewrite) ou url amigável, os projetos em Zend "dependem" de configurações no servidor em que a aplicação ficará hospedada para funcionar corretamente. É necessário que o mod_rewrite esteja habilitado e o uso de htaccess esteja disponível no apache.

Como direção para possíveis trabalhos futuros tem-se: (i) a necessidade de desenvolver testes unitários com Zend Test para as aplicações desenvolvidas com Zend Framework visando a correção de falhas não identificadas no processo de desenvolvimento; e (ii) o estudo com a finalidade de migração do Zend Framework 1 para o Zend Framework 2, visando o ganho de desempenho que a nova versão trouxe no desenvolvimento em PHP.

Referências Bibliográficas

[1] LISBOA, Flávio Gomes da Silva**. "Zend Framework – Componentes poderores para PHP",** 1ª. Ed., Editora Novatec, 2009.

[2] LISBOA, Flávio Gomes da Silva**. "Zend Framework – Desenvolvendo em PHP 5 orientado a objetos com MVC",** 1ª. Ed., Editora Novatec, 2008.

[3] ORACLE. Disponível em <http://www.oracle.com/technetwork/java/mvc-detailed-136062.html>. Acesso em: 16 de novembro de 2013

[4] Zend Framework. Disponível em: <http://framework.zend.com/manual/1.12/en/learning.quickstart.create-project.html>. Acesso em: 28 de maio de 2013

[5] School Of Net. Disponível em: <http://www.schoolofnet.com/2008/07/introducao-ao-zend-framework/>. Acesso em: 16 de novembro de 2013

[6] Zfdoc-ptbr **- "**Projeto de tradução da documentação do Zend Framework 1 para português brasileiro**"**. Disponível em: <https://code.google.com/p/zfdoc-ptbr/>. Acesso em: 11 de julho de 2013

[7] Zend. Disponível em: <http://www.zend.com/en/products/studio>. Acesso em: 11 de julho de 2013

[8] Zend Studio. Disponível em: <http://www.zend.com/en/products/studio/features>. Acesso em: 11 de julho de 2013

[9] Zend Framework. Disponível em: <http://framework.zend.com/about/faq-v1/>. Acesso em: 25 de outubro de 2013.

[10] Zend Framework. Disponível em: <http://framework.zend.com/manual/1.12/en/learning.quickstart.intro.html>. Acesso em: 25 de outubro de 2013.

[11] Entendendo a Zend Framework. Disponível em: <https://www.ibm.com/developerworks/br/library/os-php-zend1/>. Acesso em: 25 de outubro de 2013.

[12] Zend Framework. Disponível em: <http://framework.zend.com/about/>. Acesso em: 29 de outubro de 2013.