FRAMEWORK

Como Funciona a Navegação

A navegação e organização de diretórios é baseada em frameworks como CakePHP e Ruby on Rails.

Exemplo: **www.site.com.br / controlador / metodo / param1 / param2 / ...**
onde:

controlador - é uma classe situada no diretório "controlador";
metodo – é um método public do referido controlador;
param1, param2, etc - são parâmetros que o método irá receber.

Esquema básico de funcionamento

Situação: Usuário, acessando o site do EuVouPassar, quer entrar na página que explica como é o funcionamento do Aluno Vip.

O que o usuário faz:

- Digita no browser: www.euvoupassar.com.br/comofunciona/alunovip

O que acontece no código:

- O arquivo index.php, situado na raiz, é executado;

- Configurações de ambiente são executadas (declaração de constantes básicas do framework, carregamento do arquivo .ini, declaração do método autoload);

- Método despachar() da classe Controlador é executado, para interpretar a url e instanciar a classe referente ao controlador explicitado na url (no exemplo, "comofunciona");

- A classe ComoFunciona possui um método publico chamado alunoVip();

- O método alunoVip() executa as regras de negócio e depois chama o método despachar() da classe Modulo, que é a classe pai de todos os controladores (códigos comuns a mais de um controlador devem ser criados nessa classe, para reaproveitamento de código);

- O método despachar() é chamado passando-se o caminho do arquivo de visualização, situado dentro do diretório "visao". Lá existem arquivos de nome qualquer, cujo conteúdo são html que se deseja retornar ao usuário. No exemplo, poderia ser algo do tipo: $this->despachar('nomequalquer/aluno_vip.html'); , onde no arquivo aluno_vip.html, contendo as informações necessárias para se tornar um aluno vip do site.

Diretórios

- config
- controlador
- lib
- modelo
    - dados
    - excecoes
- tmp
- visao

config:

Arquivos existentes:

- autoload.php
- ambiente.php
- reporte_erros.php
- desenvolvimento.ini
- produção.ini

autoload.php:

Neste arquivo está a classe Autoload, que possui um método estático chamado load, responsável por fazer uma busca recursiva procurando a classe desejada nos seguintes diretórios: controlador, modelo, visao e lib. Caso encontre a classe desejada em algum arquivo, o mesmo será incluído através da função include_once() do PHP.

A classe Autoload é registrada para ser chamada por __autoload() através da função spl_autoload_register(array("Autoload", "load")), onde "Autoload" é o nome da nossa classe e "load" é o método a ser executado.

ambiente.php:

Neste arquivo ocorre a execução de configuração do ambiente. Uma constante chamada AMBIENTE_DEFAULT é preenchida com o nome de um dos arquivos de extensão .ini situados na pasta config (desenvolvimento e producao, por exemplo). Com isso, algumas informações relevantes são atribuídas ao framework para que ele tenha um comportamento específico de acordo com o ambiente desejado.
3 ações são executadas nesse arquivo:
A leitura dos dados contidos no arquivo .ini escolhido;
O mapeamento da url para saber qual controlador, método e parâmetros;
A inicialização automática ou não da Session (utilizando session_start);

reporte_erros.php:

Configuração de tratamento de erros (dados de configuração contidos também no ambiente sugerido na constante AMBIENTE_DEFAULT).

Exemplo: no arquivo producao.ini, display_errors está setado como false para que não seja exibido na tela para o usuário nenhum tipo de erro de execução (warning, notice, fatal error, etc); log_errors está setado como true e error_log possui o nome do arquivo onde os erros que não foram exibidos na tela do usuário sejam salvos.

Arquivos .ini :

O framework possui dois arquivos default:
- desenvolvimento.ini
- producao.ini

Nesses arquivos estão informações importantes para o comportamento do framework, como:
Exibir tempo de execução do script;
Região (ex.: 'pt_BR', 'ptb');
Timezone;
Controlador e método default (que serão executados caso não for especificado na url);
Error reporting, log de erros, display errors, arquivo .log para erros;
Dados de acesso ao banco de dados.

Você é livre para acrescentar mais arquivos .ini (no caso de outro tipo de ambiente, como um ambiente de testes por exemplo) bem como acrescentar informações aos arquivos .ini existentes. Caso precise guardar alguma outra informação que precisa estar segura, como login e senha de FTP por exemplo, poderá acrescentar facilmente ao arquivo .ini da seguinte maneira:

```
[FTP]
usuario = nome_do_usuario
senha    = senha_do_ftp
```

controlador:

Toda classe que for desempenhar o papel de controlador deve ser colocado em um arquivo dentro deste diretório. Caso ache necessário, você pode organizar seus arquivos em subdiretórios (ex.: controlador/controleslogin/Login.php).

Arquivos existentes:
- Controlador.php
- ControleException.php
- Modulo.php

Controlador.php:

Possui a classe Controlador, que pode ser considerada o cérebro do framework. Nela estão os métodos para setar informações no array registro; método despachar (chamado para executar de fato a página requerida), onde verifica se o controlador, método e parâmetros desejados são válidos, levantando exceções caso haja algum problema na identificação dos mesmos (como, por exemplo, não possuir uma classe ou método que represente o que foi desejado na url).

ControleException.php:

Possui a classe ControleException (que herda de Exception), responsável por tratar as exceções ocorridas na classe Controlador. Cada exceção é levantada com um valor específico, que é definido através de constantes da classe ControleException.

Esses valores são utilizados para identificar qual tipo de mensagem de exceção exibir na tela (por exemplo: em desenvolvimento, exibe se foi uma classe não encontrada, ou método não encontrado, ou método que foi chamado mas tem modificador private ou protected; já em produção, pode-se apenas informar uma mensagem padrão, ou até mesmo simular um erro 404).

Modulo.php:

Possui a classe Modulo, que possui atributos e métodos que são compartilhados entre os controladores que o projeto venha a possuir. Por conta disso, toda classe criada para se comportar como um controlador herda de Modulo, utilizando assim tanto os métodos de funcionamento do framework como métodos próprios que sejam projetados para serem usados entre os demais controladores.

Lembre-se: Todo controlador herda de Modulo.

Método “despachar($template)”:

Este método específico da classe Modulo é chamado nos controladores para que algum arquivo de visao seja chamado. Exemplo, caso eu tenha dentro do diretório visao um arquivo chamado exibir.php, este arquivo seria chamado de dentro de algum método de um controlador assim:

```
$this->despachar(“exibir.php”);
```

Com isso, após realizar toda a regra de negócio, o controlador chamaria a página que deseja exibir o resultado da requisição.

lib:

Diretório onde devem ser armazenadas as bibliotecas a serem utilizadas no projeto. Alguns exemplos de bibliotecas que podem ser utilizadas e que devem estar alocadas nesse diretório:

- PHPMailer;
- Editores de texto (ex.: FCKEditor, TinyMCE);
- Pager (pacote do PEAR para paginação);

modelo:

Subdiretórios: dados e excecoes.

Diretório onde estão as classes básicas, exceções personalizadas e classes de repositório. Todo objeto do sistema a ser desenvolvido deve ser representado por uma classe em um arquivo salvo neste diretório.

dados:

Subdiretório onde estão as classes para manipulação de dados do repositório. Possui dois arquivos próprios do framework:

- ConexaoPdo.php
- DadosException.php

O arquivo ConexaoPdo.php possui a classe ConexaoPdo, responsável pela conexão com o banco de dados. As informações necessárias para a conexão (como host, usuário e senha) são obtidas através do ambiente setado no arquivo config/ambiente.php.

excecoes:

Subdiretório onde estão as classes de exceções personalizadas, como por exemplo: ObjetoNaoEncontradoException. Toda classe de exceção deverá herdar da classe Exception.

visao:

Subdiretórios: css, js e imagens.

Diretório onde estão os arquivos de visualização. Imagens a serem usadas no site deverão estar dentro do diretório 'imagens', podendo ser criados subdiretórios a sua escolha; Arquivos .js deverão estar dentro do diretório 'js' e arquivos .css deverão estar dentro do diretório css. Em todos esses diretórios é possível criar subdiretórios para melhor organizar os arquivos.

Possui um arquivo que serve como template para o layout do site. Informações que se repetem ao longo do site (como cabeçalho, rodapé, menu, etc) são colocados neste arquivo, que possui uma área específica para incluir o conteúdo que é determinado pelo método despachar() dos controladores (utilizando-se <?php include($tela); ?>. O arquivo pode ter qualquer nome e deve estar alocado no diretório visao ou em algum dos seus subdiretórios.

Configuração e Segurança

Constantes do Framework:

As constantes utilizadas em uma aplicação desenvolvida com o framework devem ser declaradas no arquivo **index.php** situado na raiz do site. Por default, o framework possui algumas constantes para o uso interno do framework. São elas:

```
/**
 * Momento inicial de execução do script
 */
define('INICIO', microtime(true));

/**
 * Separador de diretórios.
 * DIRECTORY_SEPARATOR é uma constante padrão do PHP
 * que recupera o formato da barra que separa os diretórios
 */
define('DS', DIRECTORY_SEPARATOR);

/**
 * Separador de caminhos no include_path.
 * PATH_SEPARATOR é uma constante padrão do PHP
 * que recupera o separador (: no Linux e ; no Windows)
 */
define('PS', PATH_SEPARATOR);

// ------------------------------------------------------------
// Definição de constantes que representam os diretórios
/**
 * Raiz da aplicação
 */
define('RAIZ', dirname(__FILE__)); (ex.: /usr/local/apache2/...)

/**
 * Raiz da aplicação a partir do DOCUMENT_ROOT na URL
 *
 */
define('BASE', dirname($_SERVER['PHP_SELF']));
 (ex.: http://localhost/site ou http://www.site.com.br)

/**
 * Local para os arquivos gerais de configuração
 */
define('CONFIG', RAIZ . DS . 'config'); (mapeia o diretório /config)

/**
 * Local onde salvar as bibliotecas de terceiros
 */
define('LIB', RAIZ . DS . 'lib'); (mapeia o diretório /lib)

/**
 * Local dos arquivos do controlador
 */
define('CONTROLADOR', RAIZ . DS . 'controlador'); (mapeia o diretório
/controlador)
```

```
/**
 * Local onde armazenar as classes principais
 */
define('MODELO', RAIZ . DS .'modelo'); (mapeia o diretório /modelo)

/**
 * Local onde armazenar arquivos utilitários de acesso a dados
 */
define('DADOS', MODELO . DS .'dados'); (mapeia o diretório
/modelo/dados)

/**
 * Local onde armazenar as exceções de negócio
 */
define('EXCECOES', MODELO . DS .'excecoes'); (mapeia o diretório
/modelo/excecoes)

/**
 * Local dos arquivos escritos pelo próprio servidor Web
 */
define('TMP', RAIZ . DS . 'tmp'); (mapeia o diretório /tmp)

/**
 * Local dos arquivos da visão
 */
define('VISAO', RAIZ . DS . 'visao'); (mapeia o diretório /visao)
```

Conexão com banco de dados:

Os dados de acesso ao banco de dados (host, login, senha, porta) e o database a ser utilizado pela aplicação são definidos nos arquivos .ini, situados dentro do diretório /config. Existem, por default, dois arquivos: desenvolvimento.ini e producao.ini.

Para configurar o acesso ao banco de dados, basta editar os arquivos, preenchendo os campos referentes ao banco de dados. Exemplo:

[banco]
DAO = PDO
sgbd = mysql
host = localhost
usuario = *nomedousuario*
senha = *senhadousuario*
database = aplicacao_framework
porta = 5432

Arquivos .htaccess:

O que é o .htaccess?

O **.htaccess** é um arquivo de texto oculto que contém uma serie de diretivas para o **servidor Apache**. Quando um cliente solicita um arquivo ao servidor, este busca desde o diretório raiz até o subdiretório que contém o arquivo solicitado o arquivo .htaccess e aplica as regras especificadas no .htaccess ao diretório e seus subdiretórios.

E em que pode servir-me?
Em muitos casos: o .htaccess pode restringir o acesso a determinados arquivos, impedir de listar os arquivos de um diretório, redirecionar, personalizar as páginas de erro, impedir hotlinking fora de nosso site ou impedir o acesso a determinados IPs ou faixas de IP.

Uso do .htaccess no Framework:

/ - contém um .htaccess com as seguintes diretivas:

Options –Indexes

```
<IfModule mod_rewrite.c>
    RewriteEngine On

    RewriteRule ^((imagens)|(js)|(css))/(.*)$ visao/$1/$5[QSA,L]

    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^(.*)$ index.php?url=$1 [QSA,L]


</IfModule>

ErrorDocument 403 http://www.site.com.br/error/403.html (opcional)
ErrorDocument 404 http://www.site.com.br/error/404.html (opcional)
```

/config – contém um .htaccess com a diretiva “Deny from all”.
/controlador – contém um .htaccess com a diretiva “Deny from all”.
/tmp – contém um .htaccess com a diretiva “Deny from all”.
/modelo – contém um .htaccess com a diretiva “Deny from all”.

O uso do Deny from all impede o acesso ao conteúdo do diretório onde o .htaccess está.

Vantagens e Desvantagens

Vantagens:

- URL amigáveis ajudam em indexações no Google e contribuem para melhorar a usabilidade do site;
- Os ambientes (.ini) possibilitam executar o mesmo código com perfis de configuração diferentes;
- Incentivo ao uso correto de orientação a objetos, estrutura em camadas e padrões de código;
- As classes da camada de controle promovem o reuso de código através de heranças e dependências;
- Templates ajudam a flexibilizar o layout.

Desvantagens:

- Ausência de suporte a ORM;
- Uso de técnicas como Reflection e regex exigem um desempenho do servidor um pouco melhor do que os servidores comuns, em casos de grande fluxo de acessos.

Observações importantes

- O "caminho" especificado na URL não representa a organização dos diretórios dentro do diretório visao. Por exemplo: http://www.site.com.br/noticias/titulo-da-noticia.html pode ter a seguinte organização de diretórios dentro de visao: /visao/pasta_qualquer/ arquivo_qualquer.php, onde dentro do arquivo_qualquer.php está o html de visualização da notícia;

- Todos os diretórios que possuírem um arquivo .htaccess com o comando "Deny from all" não poderá ser acessado via http. Exemplo: ao tentar acessar http://www.site.com.br/config, onde estão os arquivos de ambiente, o usuário terá como resposta um erro 403 – acesso proibido.

- Toda informação sensível deve ser tratada utilizando requisições posts. Por exemplo: no gerenciador, para excluir um artigo, não se usará site.com.br/artigos/excluir/1; Se usará um botão que irá submeter um formulário, passando a ação e o id do artigo a ser excluído. No método, será chamado um método de exclusão, que verificará se a requisição foi POST e se a url que requisitou essa execução pertence ao domínio do site.

Dúvidas e informações mais detalhadas são melhores explicadas mostrando um exemplo prático.