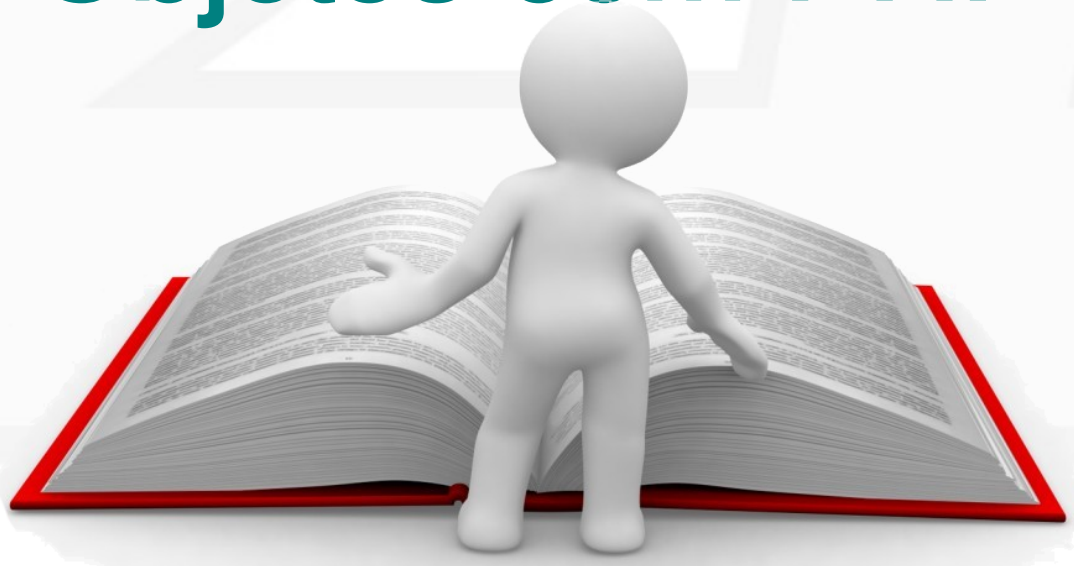
Cursos, soluções e serviços baseados em software livres e padrões abertos para ambientes de missão crítica

4LINUX
FREE SOFTWARE SOLUTIONS

Bem-vindo ao curso

Desenvolvimento Orientado a Objetos com PHP



Projeto Loja Virtual

/var/www/loja

index.php
bootstrap.php
config.php
banco.sql

Não esqueça de editar as informações de acesso ao BD!!

app
- **Controller.php**
- **View.php**
- Modules
  - Compras.php
  - Livros.php

lib
- **Banco.php**
- Carrinho.php

views
- **footer.tpl.php**
- **header.tpl.php**
- **home.tpl.php**
- Compras
  - carrinho.tpl.php
- Livros
  - alterar.tpl.php
  - inserir.tpl.php
  - livros.tlp.php
  - ver.tlp.php

Entendendo o Modelo

index.php

```
<?php

require('bootstrap.php');

new Controller();
```

Ao chamar index.php duas coisas ocorrem:

O arquivo bootstrap é lido;
A classe Controller é instanciada;

Mas como instanciamos a classe Controller se não vemos a classe aqui?

Vamos olhar o arquivo bootstrap.php

Entendendo o Modelo

bootstrap.php

```
<?php

require('config.php');

session_start();

function appload($classe) {
    $dir = dirname(__FILE__);
    $arquivo = "$dir/app/$classe.php";

    if(file_exists($arquivo)) {
        require_once($arquivo);
        return true;
    }
}

function modload($classe) {
    $dir = dirname(__FILE__);
    $arquivo = "$dir/app/modules/$classe.php";

    if(file_exists($arquivo)) {
        require_once($arquivo);
        return true;
    }
}
```

Parece que bootstrap faz um require em mais um arquivo: config.php

Veja que dentro deste arquivo temos métodos que carregam determinados arquivos caso eles existam.

Os arquivos em questão aqui são aqueles contidos em app, app/modules e lib
E um deles é o nosso Controller.php!

```
function libload($classe) {
    $dir = dirname(__FILE__);
    $arquivo = "$dir/lib/$classe.php";

    if(file_exists($arquivo)) {
        require_once($arquivo);
        return true;
    }
}

spl_autoload_register('appload');
spl_autoload_register('modload');
spl_autoload_register('libload');
```

Entendendo o Modelo

config.php

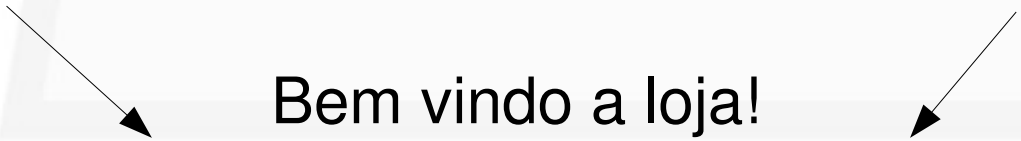
```
<?php

$config = array(
    'driver' => 'mysql',
    'host' => 'localhost',
    'database' => 'loja',
    'user' => 'root',
    'pass' => '123456'
);
```

O arquivo config.php incluído no bootstrap.php contém as informações para acesso ao banco de dados.

Depois de conhecer os arquivos iniciais do nosso modelo (index.php, config.php e bootstrap.php, vamos analisar a página inicial:

Link:
http://localhost/loja/index.php?module=livros&action=livros

Link:
http://localhost/loja/index.php?module=compras&action=carrinho



Bem vindo a loja!

Lista de produtos - Carrinho de compras

Ao invés de chamar um arquivo produtos.php por exemplo, chamaremos um module e uma ação específica para o mesmo.

Entendendo o Modelo

Controller.php

```
<?php

class Controller {
     public $view;
     public $livros;
     public $compras;

     public function __construct() {
          $this->view = new View;
          $this->livros = new Livros;
          $this->compras = new Compras;

          if(isset($_GET['module']) && isset($_GET['action'])) {
               $module = $_GET['module'];
               $action = $_GET['action'];

               if(isset($_GET['id'])) {
                    $data = $this->$module->$action($_GET['id']);
               } else {
                    $data = $this->$module->$action();
               }

               $this->view->load("$module/$action", $data);
          } else {
               $this->view->load('home');
          }
     }
}
```

Mas quem determina que, através da indicação de um module e action específicos - enviados via get - algo será mostrado na tela?

São feitas verificações para saber se todos os parâmetros estão sendo recebidos via get. Afinal, nem sempre sabemos o que será enviado pela URL.

O Controller fica então encarregado em chamar a view de acordo com o module e action passados via GET.

Entendendo o Modelo

View.php

```
<?php

class View {
    public function load($file, $data = null) {
        include("views/header.tpl.php");
        include("views/$file.tpl.php");
        include("views/footer.tpl.php");
    }
}
```

Percebam que $data dependerá se um id for enviado também na URL.

Como isso nem sempre será verdade, colocamos = null para não forçarmos que o método espere por este parâmetro.

Podemos ter diversas views diferentes. Neste caso temos apenas uma, mas subdividida em outros arquivos. Isso facilita a manutenção.

Os arquivos header.tpl.php e footer.tpl.php possuem um cabeçalho e rodapé padrão.

Já $file.tpl.php está esperando que seja carregado um arquivo com o nome ($file) passado por parâmetro.

No nosso caso, no Controller.php, resolvemos carregar apenas uma view, denominada home - $this->view->load('home'); - que consequentemente incluirá o arquivo home.tpl.php.

Entendendo o Modelo

Dicas:

Para criar um Painel de Controle para adiminstração, que tal criar uma View específica para isso? Você poderá utilizar o mesmo footer e header, moficando apenas o “centro”.

A função do Painel de Controle será conter links para que determinadas ações ocorram.

Por exemplo:

Cadastrar Cliente
Cadastrar Livro
Listar Livros (já existe)
Listar Clientes

Em Carrinho.php você já encontra métodos adicionarProduto, removerProduto e esvaziarCarrinho, mas como o próprio nome do arquivo já indica, trata-se de adicionar ou remover os produtos do carrinho (da sessão) e não do banco.

Sendo assim precisamos criar um algoritmo parecido com o que encontramos em Carrinho.php porém que faça inserções e selects no Banco.

Banco.php já está com o PDO prontinho (conectar, consultar, inserir, alterar, remover, listar, etc)

Entendendo o Modelo

Banco.php

```
<?php

class Banco {
     public static $instancia;
     protected $conexao;

     public static function instanciar() {
          if(!self::$instancia) {
               self::$instancia = new Banco;
               self::$instancia->conectar();
          }

          return self::$instancia;
     }

     protected function conectar() {
          global $config;

          $this->conexao = new PDO("{$config['driver']}:host={$config['host']};
               dbname={$config['database']}", $config['user'], $config['pass']);
          $this->conexao->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
     }

     public function executar($sql, $dados = null) {
          $statement = $this->conexao->prepare($sql);
          $statement->execute($dados);
     }
```

Entendendo o Modelo

Banco.php

```
    public function consultar($sql, $dados = null) {
        $statement = $this->conexao->prepare($sql);
        $statement->execute($dados);
        return $statement->fetchAll(PDO::FETCH_ASSOC);
    }

    public function inserir($tabela, $dados) {
        foreach($dados as $coluna => $valor) {
            $colunas[] = "`$coluna`";
            $substitutos[] = "?";
            $valores[] = $valor;
        }

        $colunas = implode(", ", $colunas);
        $substitutos = implode(", ", $substitutos);

        $query = "INSERT INTO `$tabela` ($colunas) VALUES ($substitutos)";

        $this->executar($query, $valores);
    }
```

Entendendo o Modelo

Banco.php

```
    public function alterar($tabela, $id, $dados) {
        foreach($dados as $coluna => $valor) {
            $set[] = "`$coluna` = ?";
            $valores[] = $valor;
        }

        $valores[] = $id;

        $set = implode(", ", $set);

        $query = "UPDATE `$tabela` SET $set WHERE id = ?";

        $this->executar($query, $valores);
    }

    public function remover($tabela, $id) {
        $query = "DELETE FROM `$tabela`";

        if(!empty($id)) {
            $query .= " WHERE id = ?";
        }

        $this->executar($query, array($id));
    }
```

Entendendo o Modelo

Banco.php

```
     public function listar($tabela, $campos = '*', $onde = null,
               $filtro = null, $ordem = null, $limite = null) {
          $query = "SELECT $campos FROM `$tabela`";

          if(!empty($onde)) {
               $query .= " WHERE $onde";
          }
          if(!empty($filtro)) {
               $query .= " LIKE $filtro";
          }
          if(!empty($ordem)) {
               $query .= " ORDER BY $ordem";
          }
          if(!empty($limite)) {
               $query .= " LIMIT $limite";
          }
          return $this->consultar($query);
     }

     public function ver($tabela, $campos, $onde) {
          $query = "SELECT $campos FROM `$tabela`";

          if(!empty($onde)) {
               $query .= " WHERE $onde";
          }
          return $this->consultar($query);
     }

}
```

Prática

Pessoal então mãos à obra!

A prática hoje será dedicar tempo para a criação de um Painel de Controle, para uma melhor manipulação dos dados.

Lembrem que o Modelo já possui métodos parecidos em Carrinho.php que podem serivr de base e que Banco.php já está pronto para inserir, deletar e editar dados do banco.

Precisando de uma ajuda?

A monitora e eu estaremos em pvt para ajudá-los!