**PADRÃO MVC EM PHP E FERRAMENTAS**

**Autor: Iraê César Brandão – Ultima atualização: 06/janeiro/2024**

O **PHP** (Hypertext Preprocessor) é uma linguagem de script de código aberto, especialmente adequada para o desenvolvimento web. Ela é frequentemente incorporada em HTML para adicionar funcionalidades dinâmicas a páginas da web. PHP é uma linguagem de servidor, o que significa que o código é executado no servidor web, gerando HTML que é então enviado para o navegador do usuário.

Principais características e usos do PHP:

* **Desenvolvimento Web:** PHP é amplamente utilizado para criar aplicativos web dinâmicos. Ele pode ser incorporado diretamente no código HTML ou usado em conjunto com outras tecnologias web.
* **Manipulação de Formulários:** PHP é eficaz no processamento de dados de formulários HTML. Ele pode receber dados do usuário, processá-los e enviar resultados de volta ao navegador.
* **Conexão com Bancos de Dados:** PHP oferece suporte a uma variedade de bancos de dados, como MySQL, PostgreSQL, SQLite, entre outros. Isso permite que você crie aplicações web que interagem com bancos de dados para armazenar e recuperar informações.
* **Controle de Sessão e Cookies:** PHP facilita a implementação de recursos de controle de sessão e o uso de cookies para rastrear informações do usuário entre as solicitações.
* **Manipulação de Arquivos:** PHP fornece funções para manipular arquivos no servidor, como leitura, gravação, exclusão e upload de arquivos.
* **Criação de APIs:** Pode ser usado para desenvolver APIs (Interface de Programação de Aplicações) que permitem a comunicação entre diferentes sistemas.
* **Frameworks Web:** Existem vários frameworks PHP, como Laravel, Symfony e CodeIgniter, que facilitam o desenvolvimento de aplicativos web escaláveis e organizados seguindo o padrão MVC.
* **Geração de Conteúdo Dinâmico:** PHP é frequentemente usado para criar páginas web dinâmicas, onde o conteúdo pode ser personalizado com base em variáveis, dados do usuário ou outras condições.

Em resumo, PHP é uma linguagem versátil e popular para o desenvolvimento web, oferecendo ferramentas poderosas para criar uma ampla variedade de aplicativos dinâmicos e interativos.

**Padrão de Arquitetura MVC**

O MVC (Model-View-Controller) é um padrão arquitetural de design utilizado no desenvolvimento de software. Ele divide uma aplicação em três componentes principais:

1. **Model (Modelo):** Responsável pela lógica de negócios e manipulação dos dados. Representa o estado da aplicação e responde a consultas sobre esse estado.
2. **View (Visão):** Encarregada da apresentação e exibição da interface do usuário. Recebe dados do Modelo e os apresenta de forma compreensível.
3. **Controller (Controlador):** Gerencia a entrada do usuário, traduzindo-a em comandos para o Modelo ou a Visão. Responde às ações do usuário e atualiza o Modelo conforme necessário.

A separação dessas responsabilidades facilita a manutenção, escalabilidade e reutilização de código em aplicações, promovendo uma estrutura organizada e modular. O MVC é amplamente utilizado em desenvolvimento web e aplicações GUI (Interface Gráfica do Usuário).

**Ferramentas Utilizadas**

Vamos abordar as ferramentas para a elaboração e como elas se encaixam na construção de um aplicativo usando o padrão MVC (Model-View-Controller) com PHP.

1. **VSCode (Visual Studio Code):**

* **Função:** O VSCode é um ambiente de desenvolvimento integrado (IDE) popular e poderoso. Ele fornece recursos avançados de edição de código, depuração, controle de versão e integração com várias linguagens de programação, incluindo PHP.
* **Uso no MVC:** O VSCode é uma escolha comum entre os desenvolvedores PHP devido à sua extensibilidade. Você pode instalar extensões específicas para PHP, como "PHP IntelliSense" e "PHP Debug", que facilitam a escrita de código, fornecem sugestões e oferecem suporte à depuração.

1. **XAMPP (ou outros servidores web):**

* **Função:** XAMPP é um pacote que inclui Apache (servidor web), MySQL (banco de dados), PHP e Perl. Ele simplifica a configuração de um ambiente de desenvolvimento local para facilitar o desenvolvimento de aplicativos web.
* **Uso no MVC:** Um servidor web, como o Apache fornecido pelo XAMPP, é essencial para executar e testar aplicativos PHP localmente. O Apache lida com as solicitações HTTP e executa scripts PHP, permitindo que você teste e depure seu código antes de implantá-lo em um ambiente de produção.

1. **Navegadores Web:**

* **Função:** Navegadores, como Chrome, Firefox, ou Edge, são usados para visualizar e interagir com aplicativos web. Eles interpretam o HTML, CSS e JavaScript gerados pelo servidor, exibindo a interface do usuário e permitindo a interação do usuário.
* **Uso no MVC:** Os navegadores são a interface final para os usuários. O MVC no contexto web divide a aplicação em três componentes principais: Model (lógica de negócios e acesso a dados), View (apresentação e interface do usuário) e Controller (gerenciamento de solicitações e interação entre Model e View). O PHP, no lado do servidor, é frequentemente usado para implementar o Controller e o Model, enquanto a View é geralmente composta de HTML, CSS e, às vezes, JavaScript. O navegador exibe a View e interage com o Controller por meio de solicitações HTTP.

1. **MySQL Workbench**

O MySQL Workbench é uma ferramenta gráfica popular para administrar bancos de dados MySQL, mas sua utilização não está diretamente relacionada ao padrão MVC (Model-View-Controller). O MVC é um padrão arquitetural de design que separa a lógica de negócios (Model), a apresentação (View) e o controle (Controller) em uma aplicação.

No contexto do desenvolvimento web com PHP e MySQL, o MySQL Workbench pode ser utilizado para tarefas relacionadas ao banco de dados, como criar, modificar e gerenciar tabelas, executar consultas SQL, e realizar outras operações administrativas.

Aqui está como o MySQL Workbench pode se relacionar com o desenvolvimento web e o padrão MVC:

1. **Model (Modelo):**

* O MySQL Workbench pode ser usado para projetar e criar o esquema do banco de dados (tabelas, relacionamentos, etc.), que faz parte do componente Model no padrão MVC. É aqui que você define a estrutura de dados que será manipulada pela lógica de negócios.

1. **Controller (Controlador):**

* O Controller, na lógica do PHP, pode interagir com o banco de dados para realizar operações como inserção, atualização, exclusão e recuperação de dados. O MySQL Workbench não está diretamente envolvido nessa parte, mas pode ser usado para entender e verificar os dados diretamente no banco de dados.

1. **View (Visão):**

* A View, geralmente composta por arquivos HTML e PHP, exibe dados obtidos do Model. A interação com o banco de dados é geralmente tratada pelo Controller. O MySQL Workbench não está diretamente envolvido na criação da View.

No desenvolvimento PHP com o padrão MVC, o Controller recebe solicitações do navegador, interage com o Model para obter ou modificar dados e atualiza a View para refletir essas alterações. Ferramentas como o VSCode, servidores web como o XAMPP e navegadores são essenciais para um fluxo de trabalho eficiente durante o desenvolvimento e teste de aplicativos web em PHP. Utilizamos o MySQL Workbench para tarefas relacionadas ao banco de dados (particularmente MySQL), a sua utilização não está diretamente vinculada à implementação do padrão MVC. Ele é mais frequentemente usado na fase de modelagem e administração do banco de dados, enquanto a implementação do MVC em PHP ocorre no código PHP, muitas vezes utilizando a extensão MySQLi ou PDO para interagir com o banco de dados a partir do código.

**Arquitetura (Organização e Hierarquia)**

A organização de pastas (hierarquia) em um projeto MVC (Model-View-Controller) em PHP pode variar de acordo com as preferências da equipe ou as convenções adotadas pelo framework utilizado (se houver). Vou fornecer um exemplo simples sem a inclusão de um framework, onde as pastas são organizadas manualmente.

Suponha que você tenha um projeto chamado "MeuProjeto" e está construindo um sistema de gerenciamento de tarefas. Aqui está uma possível hierarquia de pastas:

MeuProjeto/

|-- app/

| |-- Controllers/

| | |-- TaskController.php

| |

| |-- Models/

| | |-- Task.php

| |

| |-- Views/

| |-- task/

| |-- index.php

| |-- show.php

| |-- create.php

| |-- edit.php

|

|-- public/

| |-- index.php

| |-- css/

| | |-- style.css

| |

| |-- js/

| |-- script.js

|

|-- config/

| |-- database.php

|

|-- vendor/

| |-- (dependências e bibliotecas de terceiros)

|

|-- .gitignore

|-- composer.json

|-- README.md

**Aqui está uma breve descrição de cada pasta:**

1. **app/**: Contém a lógica da aplicação seguindo o padrão MVC.
   * **Controllers/**: Armazena os controladores responsáveis por gerenciar as solicitações do usuário e interagir com o modelo e a visão.
     + **TaskController.php**: Exemplo de um controlador para manipular tarefas.
   * **Models/**: Armazena as classes que representam a lógica de negócios e a interação com o banco de dados.
     + **Task.php**: Exemplo de um modelo que representa uma tarefa.
   * **Views/**: Contém os arquivos de interface do usuário (HTML, PHP).
     + **task/**: Organiza as visualizações relacionadas a tarefas.
       - **index.php**: Página de listagem de tarefas.
       - **show.php**: Página de exibição detalhada de uma tarefa.
       - **create.php**: Página de criação de uma nova tarefa.
       - **edit.php**: Página de edição de uma tarefa existente.
2. **public/**: Contém os arquivos públicos acessíveis diretamente pelo navegador.
   * **index.php**: Ponto de entrada da aplicação.
   * **css/**: Estilos CSS.
     + **style.css**: Arquivo de estilo principal.
   * **js/**: Scripts JavaScript.
     + **script.js**: Arquivo de script principal.
3. **config/**: Configurações da aplicação.
   * **database.php**: Configurações de conexão com o banco de dados.
4. **vendor/**: Pasta para armazenar dependências e bibliotecas de terceiros (pode ser gerada pelo Composer).
5. **.gitignore**: Lista de arquivos e pastas a serem ignorados pelo controle de versão Git.
6. **composer.json:** Arquivo de configuração do Composer para gerenciar as dependências do projeto.
7. **README.md:** Documentação do projeto.

Este é apenas um exemplo básico. Em projetos maiores, você pode considerar a utilização de namespaces, autoloaders e outras práticas recomendadas para organizar o código de forma mais eficiente. Além disso, ao usar frameworks como Laravel, Symfony ou CodeIgniter, a estrutura de pastas pode ser significativamente diferente, seguindo as convenções desses frameworks.

**CONVENÇÕES E FORMAS E NOMENCLATURAS**

As convenções de codificação são conjuntos de diretrizes para cada linguagem de programação específica com recomendações sobre diferentes aspectos do desenvolvimento de software nessa linguagem, incluindo estilo de codificação, melhores práticas e métodos.

**O CamelCase**: é uma convenção de nomenclatura utilizada em programação para compor palavras ou frases em que cada palavra começa com uma letra maiúscula, sem espaços ou pontuações entre elas. O nome "CamelCase" deriva do fato de que as letras maiúsculas no meio das palavras se assemelham às corcovas de um camelo.

Existem duas formas principais de CamelCase:

**Upper Camel Case (Pascal Case):**

* A primeira letra de cada palavra é maiúscula, incluindo a primeira palavra.

Exemplo: **UpperCamelCase**, **NomeDaClasse**, **FuncaoDeExemplo**.

**Lower Camel Case:**

* A primeira letra da primeira palavra é minúscula, e as primeiras letras subsequentes de cada palavra são maiúsculas.

Exemplo: **lowerCamelCase**, **nomeDaVariavel**, **funcaoDeExemplo**.

O CamelCase é frequentemente utilizado em várias linguagens de programação, incluindo PHP, JavaScript, Java, C#, e outras. Algumas das razões para usar o CamelCase incluem:

* **Legibilidade:** O CamelCase pode melhorar a legibilidade do código, especialmente em casos em que o nome da variável ou método é composto por várias palavras.
* **Padrão da Indústria:** Muitos padrões e diretrizes de codificação em diferentes comunidades de desenvolvimento adotam o CamelCase como uma prática comum.
* **Convenção em Frameworks:** Muitos frameworks e bibliotecas em diversas linguagens adotam o CamelCase para consistência em seus métodos, classes e variáveis.

As convenções para dar nomes a pastas, classes, controllers, views e models podem variar dependendo do framework que você está utilizando ou das práticas adotadas em sua equipe de desenvolvimento. No entanto, existem algumas convenções comuns que são frequentemente seguidas, especialmente em frameworks populares como Laravel, Symfony, CodeIgniter, entre outros. Aqui estão algumas sugestões de convenções:

1. **Pastas:**
   * **Controllers:**
   * Muitas vezes, a pasta de controladores é chamada de "Controllers" ou "Controller".
   * Os nomes dos controladores muitas vezes são no plural, refletindo a ideia de gerenciamento de um conjunto de entidades.

Exemplo: UsersController, TasksController.

* + **Models:**
* A pasta de modelos pode ser chamada de "Models" ou "Model".
* Os nomes dos modelos frequentemente representam uma entidade singular e podem estar no singular.

Exemplo: User, Task.

* + **Views:**
  + A pasta de visualizações pode ser chamada de "Views" ou "View".
  + Os nomes dos arquivos de visualização podem refletir a ação ou recurso, muitas vezes no singular.

Exemplo: user/index.php, task/show.php.

Composição:

app/

|-- Controllers/

|-- Models/

|-- Views/

1. **Classes:**
   * **Controllers:** Os controladores geralmente têm nomes que refletem a entidade ou recurso que estão manipulando, seguido do termo "Controller".

Exemplo:

// Nome do arquivo: Task.php

class Task {

// ...

}

* + **Models:** Os modelos geralmente têm nomes que representam a entidade de dados que estão manipulando.

Exemplo:

// Nome do arquivo: TaskController.php

class TaskController {

// ...

}

1. **Views:**

* O nome dos arquivos de visualização geralmente reflete a ação ou propósito da visualização. Por exemplo, para um controlador chamado **TaskController**, você pode ter arquivos como **index.php**, **show.php**, **create.php**, **edit.php**, etc., na pasta **Views/Task/**.

1. **Controllers e Actions (Métodos dos Controllers):**

* Os métodos em um controlador muitas vezes têm nomes que refletem as ações que estão realizando. Por exemplo, para um controlador de tarefas (**TaskController**), você pode ter métodos como **index()**, **show()**, **create()**, **edit()**, etc.

Exemplo:

class TaskController {

public function index() {

// ...

}

public function show() {

// ...

}

public function create() {

// ...

}

public function edit() {

// ...

}

}

Essas são apenas convenções comuns e não regras estritas. Ao trabalhar com um framework específico, é sempre uma boa prática consultar a documentação do framework para entender as convenções específicas que ele adota. Além disso, a consistência dentro de sua equipe de desenvolvimento é fundamental, independentemente das convenções específicas escolhidas.

Além do CamelCase, existem outras convenções de nomenclatura comumente utilizadas em PHP e no desenvolvimento em geral. Aqui estão algumas delas:

* **Snake Case:**
  + Todas as palavras são minúsculas, e espaços entre palavras são substituídos por underscores.

Exemplo: **nome\_do\_metodo**, **variavel\_nomeada**, **nome\_da\_funcao**.

* **Kebab Case (ou Hyphen Case):**
  + Semelhante ao Snake Case, mas com hífens em vez de underscores.

Exemplo: **nome-do-metodo**, **variavel-nomeada**, **nome-da-funcao**.

* **Pascal Case (ou Upper Camel Case):**
  + Semelhante ao Camel Case, mas o primeiro caractere também é maiúsculo.

Exemplo: **NomeDoMetodo**, **ClassePrincipal**, **NomeDaFuncao**.

* **Screaming Snake Case (ou Constant Case):**
  + Todas as letras são maiúsculas, e espaços entre palavras são substituídos por underscores.

Exemplo: **NOME\_DA\_CONSTANTE**, **VALOR\_MAXIMO**, **CONFIGURACAO\_PADRAO**.

* **Hungarian Notation:**
  + Prefixa o nome da variável com um ou mais caracteres indicando o tipo ou propriedade da variável.

Exemplo: **strNome**, **intIdade**, **boolAtivo**.

* **Lisp Case:**
  + Semelhante ao Kebab Case, mas usa apenas letras minúsculas.

Exemplo: **nome-da-funcao**, **variavel-de-exemplo**.

Cada convenção tem suas próprias vantagens e é escolhida com base nas preferências da equipe ou nas diretrizes do projeto. A consistência é fundamental para facilitar a leitura e manutenção do código. Em muitos projetos e frameworks PHP, Camel Case e Snake Case são amplamente adotados para nomes de classes e métodos, enquanto Snake Case é comumente usado para nomes de variáveis e colunas de banco de dados.

**REFERÊNCIAS**

DEVMEDIA. **A arquitetura MVC no desenvolvimento em PHP**. Disponível em: <<https://www.devmedia.com.br/a-arquitetura-mvc-no-desenvolvimento-em-php/2312>>. Acesso em: 06.Jan.2024.

DEVMEDIA. **Linguagem PHP.** Disponível em: <<https://www.devmedia.com.br/guia/linguagem-php/38780>>. Acesso em 06.Jan.2024.

DEVMEDIA. **Padrão MVC - Java Magazine**. Disponível em: <<https://www.devmedia.com.br/padrao-mvc-java-magazine/21995>>. Acesso em: 06.Jan.2024.

GUIA DE CARREIRA. **Entenda o que é PHP e aprenda a usar no desenvolvimento web.** Disponível em: <<https://www.guiadacarreira.com.br/blog/php>>. Acesso em: 06-Jan.2024.

MYSQL. **MySQL Workbench**. Disponível em: <<https://www.mysql.com/products/workbench/>>. Acesso em 06.Jan.2024.

STAKEOVERFLOW. **Como e para que usar MVC no PHP?** Disponível: <<https://pt.stackoverflow.com/questions/243958/como-escolher-uma-conven%c3%a7%c3%a3o-para-nome-de-vari%c3%a1veis-e-fun%c3%a7%c3%b5es>>. Acesso em: 06.Jan.2024.

STAKEOVERFLOW. **Como escolher uma convenção para nome de variáveis e funções?** Disponível: <<https://pt.stackoverflow.com/questions/114824/como-e-para-que-usar-mvc-no-php>>. Acesso em: 06.Jan.2024.

APACHE FRIENDS. **XAMPP Apache + MariaDB + PHP + Perl**. Disponível em: <<https://www.apachefriends.org/pt_br/index.html>>. Acesso em 06.Jan.2024.

WIKIPÉDIA. **CamelCase.** Disponível em: <<https://pt.wikipedia.org/wiki/CamelCase>>. Acesso em 06.Jan.2024.

WIKIPEDIA. **PHP**. Disponível em: <<https://pt.wikipedia.org/wiki/PHP>>. Acesso em 06.Jan.2024.

**CRÉDITO DE IMAGENS**

GIFER. **Idea transparent GIF - Find on GIFER**. Disponível em: <<https://gifer.com/en/YkAw>>. Acesso em: 06.Jan.2024.

MICROSOFT- ASPNET CORE. **Overview of ASP.NET Core MVC.** Disponível em: <<https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0>>. Acesso em: 06.Jan.2024.