Capítulo 8

Introdução a PHP

"Medir progresso de programação em linhas de código é como medir o progresso da construção de um avião por seu peso."
— Bill Gates

8.1 - Libertando o HTML de suas limitações

Apesar de toda evolução do HTML5 e dos navegadores atuais, a Web ainda é um ambiente bastante restrito. O browser não executa qualquer tipo de código e coisas que às vezes são simples de escrever em outras linguagens são bastante complexas de se fazer em HTML ou JavaScript.

Por isso, todo projeto Web sério não é apenas de arquivos HTML, CSS e JavaScript, mas envolve uma **infraestrutura no servidor**.

Há muitas linguagens e servidores possíveis de serem usados. Como linguagem, se usa PHP, Java, Ruby, Python, C#. Servidores temos Apache, Tomcat, JBoss, IIS, nginx e outros.

Usamos uma linguagem no servidor para executar tarefas como gerar páginas dinamicamente com dados de um banco de dados da aplicação; enviar emails para usuários; processar tarefas complexas; garantir validações de segurança da aplicação; e muito mais.

Aqui no curso, vamos usar um pouco de PHP para entender como funciona esse processo. E, principalmente, entender como o front-end (foco do curso) se integra na prática a soluções server-side.

Java vs JavaScript

Apensar do nome, essas duas linguagens são **completamente diferentes**.

Java é um linguagem voltada mais para servidores, com bastante apelo no mundo corporativo, e mantida pela Oracle.

JavaScript é a linguagem da Web para se escrever funcionalidades dinâmicas numa página. Roda no browser.

8.2 - Como funciona um servidor HTTP

Se escrevermos um HTML simples num arquivo .html, basta abri-lo no navegador e pronto, já podemos visualizar a página. Mas quando envolvemos um servidor o processo não é tão simples.

Usamos o **protocolo HTTP** para servir páginas na Web. É por isso que todo endereço na Web começa com **http://**. Quando acessamos um endereço desses na Internet, falamos que estamos fazendo uma **requisição ao servidor**. Ou seja, pedimos que certo conteúdo seja exibido.

Por exemplo, ao acessar `http://www.caelum.com.br/apostilas` estamos conectando via HTTP ao servidor www.caelum.com.br e *requisitando* a URL **/apostilas**.

Do outro lado, existe um **servidor HTTP** esperando novas requisições que é responsável por servir o que o usuário está pedindo. Esse servidor é um programa que instalamos e fica responsável por processar as requisições.

A grande questão é que esse servidor não precisa ser algo que simplesmente lê o arquivo HTML e envia seu conteúdo para o cliente. O servidor pode **executar código** e gerar HTML na hora pro cliente, **dinamicamente**. É esse processamento de lógica dinâmica no servidor que queremos fazer com PHP.

Conheça os cursos online Alura.

8.3 - Como funciona o PHP no servidor

Ao usar PHP, podemos escolher diversos servidores compatíveis. O mais famoso de todos é o Apache, que provavelmente você irá encontrar em muitas hospedagens no dia a dia.

Mas uma novidade das últimas versões do PHP (5.4+) é que ele já vem com um **servidor embutido** simples que dispensa a instalação de um servidor adicional. É ideal para testes e para usar em desenvolvimento.

Ele é muito simples de usar. Basta abrir o terminal, entrar na pasta onde está o projeto e rodar:

```
php -S localhost:8080
```

Nesse comando, a opção **-S** indica que queremos o servidor embutido dele, o endereço **localhost** indica que vamos acessar nosso servidor localmente e o valor **8080** é a porta que o servidor vai rodar.

Depois, basta navegar na URL **http://localhost:8080/** e nosso servidor vai responder normalmente. Como nosso projeto só tem arquivos .html, veremos as páginas no navegador iguaizinhas como víamos antes, sem novidades.

Mas não precisa ser assim. Para executar código do lado do servidor com PHP, basta renomear o arquivo de .html para .php. Por exemplo: **sobre.php**. Com essa extensão, podemos agora misturar código dinâmico PHP no meio do nosso HTML.

Todo código PHP fica dentro de uma tag especial dele, pra diferenciar do código HTML:

```
<?php
   // código PHP aqui
?>
```

E podemos misturar isso com HTML normalmente. O que for tag do PHP vai ser executada pelo servidor PHP. O que não for, vai ser enviado para o browser do jeito que está.

```
<h1>HTML aqui</h1>
<?php
   // código PHP aqui
?>
```

```
<p>Mais HTML</p>
```

Vamos ver muitas coisas com PHP ao longo do curso. Por enquanto, vamos fazer um exercício que apenas mostra o ano atual no rodapé da página de sobre. Para acessar o ano atual, podemos fazer:

```
<?php
   print date('Y');
?>
```

Chamamos a função `date` do PHP passando como argumento o formato que queremos a saída. No caso, **Y** indica que queremos o ano apenas. Veja outros formatos em: http://www.php.net/date

A função `date` devolve a data mas não mostra na tela. O comando `print` pega esse valor e mostra na tela.

8.4 - Para saber mais: instalação do PHP em casa

O site oficial do PHP é o http://php.net e lá você encontra downloads e código fonte completo.

Windows

Para facilitar a instalação do PHP e dependências no Windows, existe um pacote famoso chamado **WAMP** da BitNami. Ele instala o PHP, o MySQL e o servidor Apache em um clique, além de várias dependências.

Basta fazer o download e executar o instalador:
http://bitnami.com/stack/wamp

Depois de instalado, conseguimos acessar o binário do PHP pela linha de comando através de um menu. Vá em Iniciar -> BitNami Application Stack -> Use Application Stack.

Mac e Linux

Costumam vir já com o PHP instalado. Verifique apenas se a verão é igual ou superior a 5.4, que precisamos pro curso. Se for mais antiga, consulte o gerenciador de pacotes do seu sistema para atualizar.

8.5 - Exercícios: executando o PHP

1. Nosso primeiro passo é executar o servidor local do PHP. Para isso, abra o terminal e entre na pasta do projeto:

```
$ cd Desktop/mirror-fashion/
```

Em seguida, execute o servidor PHP:

```
$ php -S 0.0.0.0:8080
```

> **Os endereços do server builtin do PHP**
>
> O argumento **-S** habilita o servidor do PHP. Como argumento, ele recebe um IP e uma porta. Quando passamos **0.0.0.0**, estamos habilitando todos os IPs da máquina; isso quer dizer que o servidor é acessível tanto na própria máquina quanto via rede. Isso é útil para testarmos nosso projeto em dispositivos móveis conectados na rede, por exemplo.

2. Abra o navegador e acesse **http://localhost:8080**. Você deverá ver a página da Mirror Fashion sendo servida pelo PHP agora.

Para testarmos nossa instalação do PHP, vamos implementar uma funcionalidade bem simples, porém muito útil. Isto é, queremos saber o ano atual dinamicamente e inserir esse valor na página. Com PHP, isso é muito simples: basta usar a função `date`.

3. **Primeiro** renomeie a extensão do arquivo **sobre.html** para **sobre.php**. Lembre que um arquivo PHP nada mais é que um HTML com instruções especiais pra rodar no servidor.

4. O texto que descreve a Mirror Fashion fala de sua fundação em 1932. Acrescente uma frase dinâmica no texto que indica há quantos anos a empresa foi fundada.

```
Fundada há <?php print date("Y"); ?> anos,...
```

5. Repare que no lugar de sair a quantidade de anos, estamos exibindo o ano atual. Para resolver isso, vamos subtrair o ano atual menos a data de fundação da Mirror Fashion:

```
Fundada há <?php print date("Y") - 1932; ?> anos,...
```

6. (opcional) A função `date` recebe como parâmetro o formato que desejamos para nossa data. Teste outros valores, como **m** ou **l**.

Consulte outros valores na documentação: http://www.php.net/date

8.6 - Reaproveitamento de código com include

Um recurso muito comum de aparecer em todo projeto é a criação de um cabeçalho e um rodapé único para o site que são aproveitados em todas as páginas. Cada página individual só muda o miolo.

O problema é que, com HTML puro, a única solução é ficar copiando e colando o código do cabeçalho em todas as páginas. Isso é muito ruim. Se um dia precisarmos mudar um item no menu do cabeçalho, temos zilhões de arquivos pra mexer.

O HTML não tem recursos pra que deixemos esse código **centralizado** em um só lugar. Existem algumas soluções que, ou são muito limitadas ou têm problemas sérios de suporte nos navegadores. Então, *do ponto de vista do HTML* não há outro jeito: precisamos copiar e colar o código em toda página.

Mas, usando uma tecnologia no servidor como PHP, podemos fazer esse "copiar e colar" dinamicamente. Criamos um arquivo .php separado que **encapsula** o código do cabeçalho. Depois, incluímos esse código em todas as páginas usando o comando `include`.

```
<?php
  include("cabecalho.php");
?>
```

Repare que o HTML final que o browser recebe é o mesmo de antes. A inclusão

do cabeçalho acontece no servidor.

8.7 - Exercícios: include

1. A partir do próximo capítulo, vamos desenvolver a página de detalhes de produto completa. Mas o primeiro passo nosso é criar a estrutura básica do arquivo. Isto inclui o doctype, tag html, head, body, title.

Crie o arquivo **produto.php** com uma base parecida com essa:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Mirror Fashion</title>

    <link rel="stylesheet" href="css/reset.css">
    <link rel="stylesheet" href="css/estilos.css">
    <link rel="stylesheet" href="css/mobile.css" media="(max-width: 939px)">
  </head>
    <body>

    </body>
  </html>
```

Repare que já incluímos algumas tags que vimos antes no curso. Em especial, o charset como UTF-8, a tag viewport para nossa página funcionar bem em mobile e os arquivos CSS de reset e estilos.

Como todas as páginas fazem parte da Mirror Fashion, é muito comum que tenham o mesmo cabeçalho. Tínhamos criado um cabeçalho bacana na **index.php** e, para termos o mesmo cabeçalho no **produto.php**, teríamos que copiar e colar o código do `<header>`.

O HTML sozinho não tem recursos muito interessantes para se reaproveitar pedaços de código entre páginas. Mas no servidor isso é *bem fácil* de resolver. Com PHP, basta usar a função `include`.

2. Primeiro, **crie** um arquivo chamado **cabecalho.php** e coloque o conteúdo do cabeçalho que temos na home com a tag `<header>`. Esta é uma boa hora para usar o cortar-e-colar:

```
<header class="container">
  <h1>
    <img src="img/logo.png" alt="Mirror Fashion">
  </h1>
```

```
  <p class="sacola">
  Nenhum item na sacola de compras
  </p>
  <nav class="menu-opcoes">
  <ul>
    <li><a href="#">Sua Conta</a></li>
    <li><a href="#">Lista de Desejos</a></li>
    <li><a href="#">Cartão Fidelidade</a></li>
    <li><a href="sobre.html">Sobre</a></li>
    <li><a href="#">Ajuda</a></li>
  </ul>
  </nav>
  </header>
```

3. Na página **produto.php**, inclua **cabecalho.php** logo no **início do body**:

```
<body>
  <?php include("cabecalho.php"); ?>
</body>
```

4. Teste a nova página acessando **http://localhost:8080/produto.php**. O cabeçalho deve aparecer incluído. Verifique o HTML da página pelo navegador.

5. Crie o arquivo **rodape.php** para fazermos a mesma coisa com o rodapé copiando o conteúdo do `<footer>` que havíamos criado na Home:

```
<footer>
  <div class="container">
  <img src="img/logo-rodape.png" alt="Logo Mirror Fashion">
  <ul class="social">
  <li><a href="http://facebook.com/mirrorfashion">Facebook</a></li>
  <li><a href="http://twitter.com/mirrorfashion">Twitter</a></li>
  <li><a href="http://plus.google.com/mirrorfashion">Google+</a></li>
  </ul>
   </div>
  </footer>
```

6. Na página **produto.php**, inclua **rodape.php** logo antes de fechar o body usando o `include` do PHP:

```
<body>
  <?php include("cabecalho.php"); ?>
  <?php include("rodape.php"); ?>
</body>
```

7. O que ganhamos fazendo o include com PHP? Qual o trabalho de editar o logo da empresa, por exemplo, se tivermos 100 páginas no site?

8. (opcional) Aplique o cabeçalho e o rodapé que acabamos de criar também na Home e na página de Sobre. Para isso, transforme esses arquivos em PHP renomeando suas extensões e use o `include`.

8.8 - Para saber mais: ainda mais flexibilidade com variáveis

Podemos passar variáveis de um arquivo para o outro durante o include. Por exemplo, a página do cabeçalho pode receber um título para imprimir no `<title>` ao invés de deixar um valor fixo.

```
<title><?php print $cabecalho_title; ?></title>
```

E na página **produto.php**, definimos a variável *antes* de dar o include:

```
<?php
  $cabecalho_title = "Produto da Mirror Fashion";
  include("cabecalho.php");
?>
```


8.9 - Exercícios opcionais: variáveis em PHP

1. Edite o arquivo **cabecalho.php** para incluir toda a estrutura inicial do arquivo HTML e não só o topo da página. Coloque desde o doctype, abertura da tag html, head, body até o header em si.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mirror Fashion</title>

    <link rel="stylesheet" href="css/reset.css">
    <link rel="stylesheet" href="css/estilos.css">
    <link rel="stylesheet" href="css/mobile.css" media="(max-width: 939px)">

    <meta name="viewport" content="width=device-width">
  </head>
```

```
  <body>

    <header class="container">
      <!-- conteúdo do header aqui -->
    </header>
```

Agora remova esse pedaço do doctype, html, head e body de dentro das páginas que fazíamos include. A ideia é que todo esse pedaço agora é reaproveitável no include e não apenas o header em si.

2. Mas algumas partes do HTML que estamos incluindo agora no cabecalho.php devem ser dinâmicas. O `<title>` por exemplo não deveria ficar fixo no include, mas deveria ser diferente para cada página.

Uma forma de resolver isso é passando variáveis entre as páginas.

Na **produto.php**, antes de incluir o cabeçalho, defina uma variável com o título:

```
<?php
  $cabecalho_title = "Produto da Mirror Fashion";
  include("cabecalho.php");
?>
```

E no **cabecalho.php**, vamos imprimir essa variável dentro do `<title>`:

```
<title><?php print $cabecalho_title; ?></title>
```

Repare como agora o título é parametrizável. Defina um título nas outras páginas que fazem uso do incinclude também (Home, Sobre etc).

3. Que outros elementos desse cabeçalho podem mudar entre páginas diferentes além do título? Implemente soluções parecidas usando variáveis pra resolver outros casos de include dinâmico.

Exemplo: cada página pode incluir um arquivo CSS próprio, com seu estilo. Não vamos querer listar todos os arquivos CSS no cabecalho.php. O melhor é cada página declarar qual arquivo .css adicional quer incluir além dos básicos.

Na **produto.php**, podemos querer incluir um **produto.css**. Vamos usar uma variável pra isso:

```
<?php $cabecalho_css = '<link rel="stylesheet" href="css/produto.css">'; ?>
```

E no **cabecalho.php** podemos incluir essa variável no meio do nosso head:

```
<?php print $cabecalho_css; ?>
```

4. Nem todas as páginas precisam de um arquivo .css extra. Do jeito que fizemos, se a variável `$cabecalho_css` não for definida, um erro acontecerá. Isso é ruim.

Uma forma de evitar é tentar imprimir a variável apenas se ela existir, e ignorar isso caso ela não esteja definida. Ou seja, a variável é opcional.

Podemos fazer isso no PHP indicando que o erro de variável não encontrada pode ser ignorado. Para isso, usamos o **@** na frente da variável:

```
<?php print @$cabecalho_css; ?>
```

CAPÍTULO ANTERIOR:

Web para dispositivos móveis

PRÓXIMO CAPÍTULO:

Progressive enhancement e mobile-first

Newsletter

Casa do Código

Twitter