



## Primeiros passos com PHP

Bem vindo ao nosso curso de PHP e MySQL. Nesse curso, veremos como fazer com que páginas estáticas fiquem dinâmicas, ou seja, sites que reagem de acordo com a ação do usuário. E para isso, utilizaremos o PHP, que é uma tecnologia que nos ajuda nessa tarefa.

Um bom exemplo são os sites das lojas virtuais, como Amazon e etc. Neles, você consegue adicionar e remover produtos do carrinho, alterar produtos, etc. E a grande graça é que todas as informações ficam salvas em servidores na Internet. Assim, qualquer pessoa pode acessar a lista de produtos sempre atualizada.

Neste curso, faremos uma loja de produtos, e o disponibilizaremos na internet. A aplicação será grande: adicionaremos produtos, categorias, os listaremos, e etc. No meio do caminho, precisaremos aprender diversos fundamentos da web. Como funciona o processo de termos um servidor e um cliente, distantes um do outro? Como vários clientes conseguem acessar o mesmo servidor?

Para começar o curso, você já deve ter instalado na sua máquina o XAMPP. O XAMPP nada mais é do que um utilitário que já instala o PHP, o Apache e o MySQL, que são todas as ferramentas que precisamos para trabalhar no curso. Se você ainda não instalou, você o instalará no primeiro exercício.

Vamos conferir se a instalação está correta. No diretório que criamos as nossas páginas (que é o diretório de instalação do XAMPP, por exemplo, `/Applications/XAMPP/xamppfiles/htdocs`), vamos criar um diretório `/loja`. Em seguida, vamos abrir esse diretório em algum editor de texto, para que possamos lidar com os diversos arquivos que criaremos ao longo do tempo no nosso curso.

Nosso primeiro arquivo se chamará `index.php`, e terá uma mensagem simples:

```
<html>
  <h1>Bem vindo!</h1>
</html>
```

Se quisermos testar, basta abriremos o browser e acessarmos: `http://localhost/loja/`. Localhost quer dizer a própria máquina, e `/loja` é justamente o sub-diretório que criamos. Poderíamos acessar o endereço de outra forma também: `http://localhost/loja/index.php`.



Vamos entender o que aconteceu. Quando você digitou um endereço no navegador, o navegador foi até o servidor, e o servidor devolveu uma resposta para o browser, que a mostrou. É possível ver o HTML que chegou no browser, por meio da opção `View -> Developer -> View Source` do Chrome.

No caso acima, ambas as URLs devolvem o mesmo conteúdo, pois o servidor é inteligente: se você faz uma requisição para uma pasta, e não especifica um arquivo exato (como um .php ou .html), ele então procura por um arquivo chamado "index.php", e devolve o conteúdo dele.

Uma outra maneira também de ver a requisição e a resposta acontecendo é usando a aba de Desenvolvedor do Chrome. `View -> Developer -> Developer Tools`. Outros navegadores também tem opções semelhantes. Se dermos um refresh na página, podemos vê-la na aba `Network`. Se olharmos ali, vemos que a requisição funcionou (voltou status 200, que significa sucesso, e entenderemos melhor mais pra frente).

Vamos agora criar a pagina que adiciona um produto: `adiciona-produto.php`. Vamos começar colocando a mensagem de cadastro com sucesso:

```
<html>
```

Produto NOME adicionado com sucesso!

</html>

Vamos abrir o browser e testar: `http://localhost/loja/adiciona-produto.php`. Ele exibe a mensagem. Vamos agora passar parâmetros pra essa página, pela própria URL. Vamos acessar o seguinte endereço: `http://localhost/loja/adiciona-produto.php?nome=carro&preco=5000`. Veja que usamos o `?` e passamos uma lista de chaves e valores: `nome=carro`, `preco=5000`, e assim por diante; separamos cada item com `&`.

Mas ao acessarmos essa página, parece que ele "descartou" o nome que passamos, pois a mensagem ainda tem o texto "NOME". É agora que começaremos a programar em PHP. Precisamos dizer na nossa página que temos uma **variável** nome, que vem do usuário. Para isso, precisamos programar em PHP. Código PHP não é código HTML, e precisamos separar bem ambos para que o servidor entenda e processe corretamente. Repare o `<?php`:

```
<html>
<?php
$nome = $_GET["nome"];
?>
Produto NOME adicionado com sucesso!
</html>
```

Veja que dentro do código PHP, declaramos uma variável "nome", usando `$`, e usamos o `$_GET[]`, que é um array, e pega os valores que foram passados pelo usuário na URL. No nosso caso, fizemos `$_GET['nome']`, porque o parâmetro enviado pela URL se chama "nome". Podemos fazer a mesma coisa para o "preco". Vamos já também imprimir a variável "nome" também no lugar da palavra "NOME":

```
<html>
<?php
$nome = $_GET["nome"];
$preco = $_GET["preco"];
?>
Produto <?php echo $nome; ?> adicionado com sucesso!
</html>
```

Quando o comando php é escrito em uma única linha, o ponto-e-vírgula é opcional. Aqui, vamos usar para manter um padrão. Vamos dar um refresh na página e ver o

resultado. O nome agora apareceu no lugar certo!

Vamos exibir o preço também:

```
<html>
<?php
$nome = $_GET["nome"];
$preco = $_GET["preco"];
?>
Produto <?php echo $nome; ?>, <?php echo $preco; ?> adicionado com sucesso!
</html>
```

Também funciona! Como exibiremos informação o tempo todo no PHP, existe um atalho que facilita: `<?= $nome; ?>`:

```
<html>
<?php
$nome = $_GET["nome"];
$preco = $_GET["preco"];
?>
Produto <?= $nome; ?>, <?= $preco; ?> adicionado com sucesso!
</html>
```

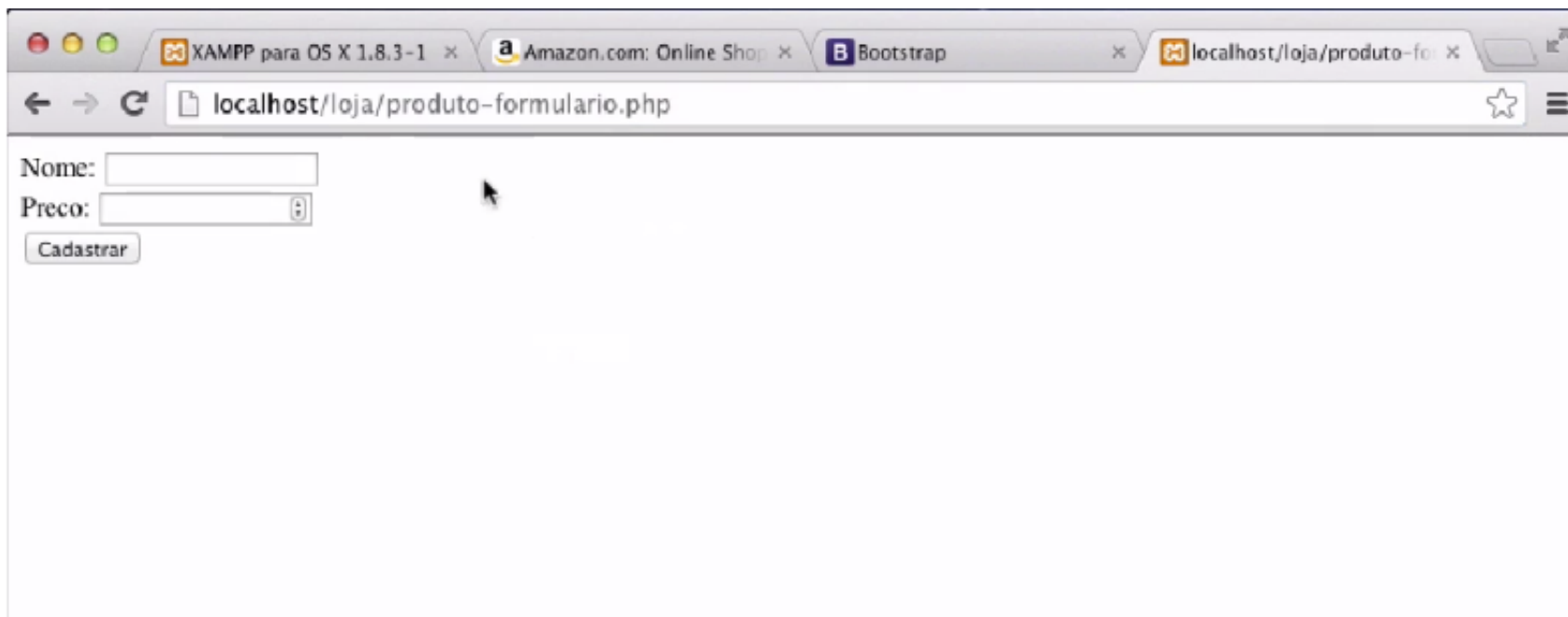
Agora vamos criar o formulário que vai mandar essas informações, afinal não queremos que o usuário digite direto na URL. No arquivo `produto-formulario.php`, colocaremos:

```
<html>
  <form>
    Nome: <input type="text" name="nome" /><br/>
    Preço: <input type="number" name="preco" /><br/>

    <input type="submit" value="Cadastrar" />
```

```
</form>  
</html>
```

Veja que demos nomes aos campos de texto, pois é através desse nome, que conseguiremos recuperar a informação lá no PHP. Se acessarmos o formulário pelo browser, temos a seguinte tela:



The screenshot shows a web browser window with the address bar displaying 'localhost/loja/produto-formulario.php'. The browser tabs include 'XAMPP para OS X 1.8.3-1', 'Amazon.com: Online Shop', 'Bootstrap', and 'localhost/loja/produto-formulario.php'. The form itself consists of two text input fields. The first is labeled 'Nome:' and the second is labeled 'Preço:'. Below these fields is a button labeled 'Cadastrar'.

Se preenchermos o formulário, e clicarmos no botão, ele vai enviar as informações! Mas repare que ele enviou para a página errada. Ele enviou para a própria página. Precisamos mudar isso, pois precisamos mandar para a página `adiciona-produto.php`. Fazemos essa mudança no form:

```
<form action="adiciona-produto.php">  
  
</form>
```

Vamos testar de novo. Abrimos o formulário e cadastramos. Agora funcionou! Os dados que foram digitados no formulário, foram submetidos para a `adiciona-produto.php`, que por sua vez pegou os valores informados pelo usuário e os exibiu. Veja que aqui tivemos 2 requisições: uma para pedir o formulário, e a resposta foi

o formulário; a outra foi o envio das informações, e a resposta foi a mensagem elegante confirmando a adição.

Mas por enquanto nosso sistema está feio. Vamos fazer uso de uma biblioteca famosa, chamada Bootstrap, que vem com um conjunto de CSS que já nos ajudam. Você pode baixá-lo em [getbootstrap.com](http://getbootstrap.com), clicando em Download Bootstrap.

Ao descompactar a biblioteca, vemos que ela é composta por 3 diretórios: css, fonts, js. Vamos copiar as 3 pastas para nosso projeto. Para usá-lo, basta importar o CSS. Vamos na nossa página principal, criar a tag HEAD, e importar o CSS do bootstrap. Além disso, vamos fazer uso das classes do bootstrap, que deixarão nosso site bonito:

```
<html>
<head>
  <title>Minha loja</title>
  <meta charset="utf-8">
  <link href="css/bootstrap.css" rel="stylesheet" />
  <link href="css/loja.css" rel="stylesheet" />
</head>

<body>
  <div class="container">

    <div class="principal">
      <h1>Bem vindo!</h1>
    </div>

  </div>

</body>
</html>
```

Veja também que colocamos o loja.css, que será o css da nossa loja. Vamos criá-lo, pois precisamos escrever a classe "principal":

```
body {
```

```
padding-top: 50px;
}

.principal {
padding: 40px 15px;
text-align: center;
}
```

Se abirmos o index.php no browser, vemos que a fonte já está mais elegante, o texto centralizado, e etc. Aos poucos vamos usando mais coisas do Bootstrap.

Vamos levar o mesmo layout pra outra página:

```
<html>
<head>
  <title>Minha loja</title>
  <meta charset="utf-8">
  <link href="css/bootstrap.css" rel="stylesheet" />
  <link href="css/loja.css" rel="stylesheet" />
</head>

<body>
  <div class="container">

    <div class="principal">

      <?php
$nome = $_GET["nome"];
$preco = $_GET["preco"];
?>
      Produto <?= $nome; ?>, <?= $preco; ?> adicionado com sucesso!

    </div>
```

```
</div>

</body>
</html>
```

Mas dá pra melhorar ainda mais, já que o Bootstrap tem bastante coisa legal. Vamos pegar a mensagem de sucesso e colocar num parágrafo com a classe "alert-success":

```
<p class="alert-success">
    Produto <?= $nome; ?>, <?= $preco; ?> adicionado com sucesso!
</p>
```

Vamos fazer a mesma coisa agora na página de formulário, e colocar um H1 para indicar que é um formulário:

```
<html>
<head>
    <title>Minha loja</title>
    <meta charset="utf-8">
    <link href="css/bootstrap.css" rel="stylesheet" />
    <link href="css/loja.css" rel="stylesheet" />
</head>

<body>
    <div class="container">

        <div class="principal">

            <h1>Formulário de cadastro</h1>
            <form action="adiciona-produto.php">
                Nome: <input type="text" name="nome" /><br/>
                Preço: <input type="number" name="preco" /><br/>
```



```
        <input type="submit" value="Cadastrar" />
    </form>

</div>

</div>

</body>
</html>
```

O formulário também já está melhor. Mas o problema é que nesses arquivos temos um monte de copy-and-paste. Todo copiar e colar é ruim, pois se precisarmos mudar algo, precisaremos mudar em todos, e o trabalho será muito grande!

Vamos então separar esse código repetido e colocar em um único arquivo, começando pela parte de cima, no `cabecalho.php`:

```
<html>
<head>
    <title>Minha loja</title>
    <meta charset="utf-8">
    <link href="css/bootstrap.css" rel="stylesheet" />
    <link href="css/loja.css" rel="stylesheet" />
</head>

<body>
    <div class="container">

        <div class="principal">
```

A mesma coisa para a parte de baixo, no `rodape.php`:

```
</div>
```

```
</div>

</body>
</html>
```

Vamos agora fazer uso desses arquivos que criamos. Nos arquivos anteriores, vamos jogar fora o "código repetido" e incluir ambos os PHPs que criamos:

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

<h1>Formulário de cadastro</h1>
<form>
    Nome: <input type="text" name="nome" /><br/>
    Preço: <input type="number" name="preco" /><br/>

    <input type="submit" value="Cadastrar" />
</form>

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

Veja só como agora está melhor. Se mudarmos o cabeçalho ou rodapé, mudaremos em um único lugar!

Nesse capítulo então aprendemos que o mundo web funciona através de requisições e respostas. O servidor processa a requisição e envia uma resposta de volta ao browser. Aprendemos também que para programar PHP no meio do HTML, basta abrir a tag PHP. Aprendemos também a reaproveitar código, usando a tag `include`.

Sugira uma correção

Ver video

Começar exercícios

