02

Cabeçalho e rodapé: customizando o Code Igniter

Olhando a página do produto ainda temos algo de estranho: o design fica razoável e lembra as melhorias de css que utilizamos, mas fizemos isso em diversas páginas através de um copy e paste de cabeçalho (e possivelmente de um rodapé). Vamos criar então um cabeçalho, no diretório `views` criamos um `cabecalho.php`:

```
<html>
<head>
    <link rel="stylesheet" href="<?= base_url("css/bootstrap.css") ?>">
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
    <div class="container">

        <?php if($this->session->flashdata("success")) : ?>
            <p class="alert alert-success"><?= $this->session->flashdata("success") ?></p>
        <?php endif ?>

        <?php if($this->session->flashdata("danger")) : ?>
            <p class="alert alert-danger"><?= $this->session->flashdata("danger") ?></p>
        <?php endif ?>
```

Criamos também o `views/rodape.php`:

```
    </div>
</body>
</html>
```

Toda página nossa estará dentro do cabeçalho e do rodapé. Como fazer o Code Igniter colocar isso em todas nossas páginas? Primeiro vamos no `Produtos` que é o controller:

```
    public function index()
    {
        $this->load->model("produtos_model");
        $produtos = $this->produtos_model->buscaTodos();

        $dados = array("produtos" => $produtos);
        $this->load->helper(array("currency"));
        $this->load->view("produtos/index.php", $dados);
    }
```

Além de carregar o `index.php` vamos falar para carregar o cabeçalho e o rodapé:

```
    public function index()
    {
        $this->load->model("produtos_model");
        $produtos = $this->produtos_model->buscaTodos();
```

```
        $dados = array("produtos" => $produtos);
        $this->load->helper(array("currency"));
        $this->load->view("cabecalho.php");
        $this->load->view("produtos/index.php", $dados);
        $this->load->view("rodape.php");
    }
```

Removemos da nossa página index todo o código que está no cabeçalho e rodapé. Funciona, mas temos que fazer isso em todas as páginas? Isto é, no `mostra` do `Produtos`, no `formulario` do `Produtos` ... copy e paste:

```
        $this->load->view("cabecalho.php");
        $this->load->view("produtos/" COM O RESTO DO NOME AQUI ".php", $dados);
        $this->load->view("rodape.php");
```

Mas estamos copiando e colando código muito parecido, a única coisa que muda é o nome da página central. Seria possível chamar o método `view` uma única vez? Mas calma lá, o método `view` foi criado pelo Code Igniter, não quero mudar o código fonte dele, então gostaria de poder chamar algo como:

```
        $this->load->template("produtos/index.php", $dados);
```

E ai a minha função customizada `template` faça o load das três views. Quem carrega as páginas no Code Igniter é o `Loader` e o que eu quero então é fornecer o meu `Loader`, uma extensão para o Code Igniter.

No nosso diretório `core` crio o meu `MY_Loader.php` (o nome é padrão, importante). Colocamos uma classe que estende o `CI_Loader`:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class MY_Loader extends CI_Loader {

}
```

Dentro dele posso definir minha função `template`:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class MY_Loader extends CI_Loader {

    public function template($nome, $dados = array()) {
    }

}
```

Repare que definimos a array de dados como opcional, se ela não for passada ela é uma array vazia. Agora chamamos a função `view` em nós mesmos três vezes:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class MY_Loader extends CI_Loader {
```

```
    public function template($nome, $dados = array()) {
        $this->view("cabecalho.php");
        $this->view($nome, $dados);
        $this->view("rodape.php");
    }

}
```

Pronto, em todo lugar de nossa aplicação que desejamos usar o template basta usarmos agora:

```
$this->load->template('minha_pagina', $dados);
```

Com isso completamos a customização de um dos componentes do Code Igniter - podemos alterar diversos deles. Vimos nesse curso como criar uma aplicação do zero com um framework PHP, acessar o banco, utilizar controllers, modelos, views, emails, helpers, migrations e muito mais.