09

Importanto e refatorando o formulário para utilizar Bootstrap

Nosso formulário ainda está feinho. Podemos estilizar usando um pacote CSS pronto, o Bootstrap! Primeiro baixe o Bootstrap clicando aqui (https://github.com/alura-cursos/play-framework/archive/bootstrap.zip) e extraia na pasta **public**.

Agora vamos importar os estilos. Faremos isso editando o arquivo **main.scala.html**. Nós falamos dele ao criar o formulário inicialmente, mas nunca entendemos o que ele faz. Então aí está! Olhando seu conteúdo, podemos ver que ele é um tipo de envelope, um wrapper, que inclúi diversas informações importantes para páginas web como as tags `html`, `head` e `body`.

Podemos ver alguns comentários explicando o que cada elemento representa. Se quiser dê uma lida. Após apagar os comentários, vamos fazer uma grande mudança aqui. Primeiro, vamos remover os imports já existentes e substituir pelos do Bootstrap, incluindo tanto o CSS quanto o Javascript:

```
@(title: String)(content: Html)
<!DOCTYPE html>
<html>
    <head>
        <title>@title</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" media="screen" href="@routes.Assets.versioned("bootstrap/css/boot
        <link rel="shortcut icon" type="image/png" href="@routes.Assets.versioned("images/favic
    </head>
    <body>
        @content
        <script type="text/javascript" src="@routes.Assets.versioned("bootstrap/js/jquery.min.js
        <script type="text/javascript" src="@routes.Assets.versioned("bootstrap/js/bootstrap.mi
    </body>
</html>
```

Para acessar os arquivos da pasta **public**, utilizamos a última rota do arquivo de rotas, uma rota padrão criada pelo Play!: `@routes.Assets.versioned()`. Com ela, você pode acessar qualquer arquivo da pasta.

O formulário fica um pouco mais estilizado, já que assume alguns dos padrões do Bootstrap, mas ainda podemos melhorar. Para descolar o conteúdo do topo da página, podemos fazer um cabeçalho (que pode servir de menu de navegação). Vamos envolver também o conteúdo do `main` em um `container`, para que fique mais centralizado.

```
<body>
    <header class="navbar navbar-default">
        <div class="container" role="presentation">
            <ul class="nav navbar-nav navbar-right">
                <li><a href="@routes.ProdutoController.formularioDeNovoProduto">Novo produto</a
            </ul>
        </div>
    </header>
    <main class="container">
        @content
    </main>
```

```
    <script ... type="text/javascript" src="@routes.Assets.versioned("bootstrap/js/jquery.min.j
    <script type="text/javascript" src="@routes.Assets.versioned("bootstrap/js/bootstrap.min.js"
</body>
```

Agora para deixar o formulário de acordo com a estilização do Bootstrap, vamos utilizar as classes de formulário para deixá-lo mais elegante? Faremos isso utilizando uma biblioteca que serve de envelope para o Bootstrap, e para isso temos que incluí-la no arquivo `build.sbt`, adicionando a linha `"com.adrianhurt" % "play-bootstrap_2.11" % "1.0-P25-B3"` na lista de dependências. Faça um `reload` no console do *Activator* e suba o projeto de novo utilizando o comando `~run .`

Precisamos então alterar o formulário para utilizar o envelope do Bootstrap. Ao invés de importar o pacote `helper`, importaremos o `b3.vertical.fieldConstructor` e alteramos as chamadas para utilizar a nova biblioteca:

```
@(formulario: Form[Produto])
@import b3.vertical.fieldConstructor
@main("Cadastro de produto") {
    @b3.form(routes.ProdutoController.salvaNovoProduto) {
        <h1>Cadastrar novo produto</h1>
        @b3.text(formulario("titulo"), '_label -> "Título")
        @b3.text(formulario("codigo"), '_label -> "Código")
        @b3.text(formulario("tipo"), '_label -> "Tipo")
        @b3.textarea(formulario("descricao"), '_label -> "Descrição")
        @b3.number(formulario("preco"), '_label -> "Preço")
        @b3.submit('class -> "btn btn-primary") { Cadastrar }
    }
}
```

Repare que agora somos obrigados a incluir os rótulo explicitamente com uma sintaxe semelhante à dos atributos do ajudante nativo. Porém temos suporte pronto ao `<input type="number">` e um suporte relativo ao botão de submissão!