02

Layouts com o Twitter Bootstrap

Layout para a aplicação

Agora que já sabemos como utilizar o Asp.Net MVC, chegou a hora de melhorar a aparência da aplicação. Para isso utilizaremos uma biblioteca css chamada twitter bootstrap que pode ser baixada no site http://getbootstrap.com (http://getbootstrap.com) nesse curso utilizaremos a versão 3.1.1 que pode ser baixada nesse link https://github.com/twbs/bootstrap/releases/download/v3.1.1/bootstrap-3.1.1-dist.zip (https://github.com/twbs/bootstrap/releases/download/v3.1.1/bootstrap-3.1.1-dist.zip).

Depois do download do bootstrap, precisamos colocá-lo dentro da aplicação. Arquivos de conteúdo estático como css e imagens são geralmente colocados dentro de uma pasta chamada `Content` do projeto, vamos então criar essa nova pasta e dentro dela, criaremos a pasta `Css` que conterá todos os arquivos css que serão utilizados. Dentro dessa pasta, cole o arquivo `bootstrap.css` que pode ser encontrado dentro do zip do twitter bootstrap, além disso, coloque nessa pasta também o arquivo de estilo do site que pode ser baixado nessa url: Onde?

Vamos começar aplicando o layout na lista de produtos da aplicação. Abra o arquivo `Views/Produto/Index.cshtml` e dentro da tag head, vamos importar os arquivos css utilizando a tag `link`:

```
<link rel="stylesheet" href="url do css" />
```

No `href` do `link`, precisamos passar a url do arquivo css dentro da pasta `Content/Css` do projeto. No Razor, podemos conseguir o endereço da pasta da aplicação utilizando o símbolo `~` no código do `href`. Para chegarmos ao `bootstrap.css`, precisamos, a partir da pasta do projeto, acessar a pasta `Content/Css`, então a url fica da seguinte forma: `~/Content/Css`. Agora podemos utilizar essa url na tag `link`:

```
<link rel="stylesheet" href="~/Content/Css/bootstrap.css"/>
<link rel="stylesheet" href="~/Content/Css/Site.css"/>
```

Essa tag link é importada dentro do head da página:

```
<head>
  <link rel="stylesheet" href="~/Content/Css/bootstrap.css"/>
  <link rel="stylesheet" href="~/Content/Css/Site.css"/>
</head>
```

Agora precisamos modificar o código do `body` da página para que ele utilize o twitter bootstrap para fazer o layout. Começaremos com um menu superior onde poderemos acessar as diferentes páginas da aplicação. Dentro da tag `body`, colocaremos uma `div` com a classe `container` e dentro dessa tag, colocaremos um novo `div` para o cabeçalho da página:

```
<body>
  <div class="container">
    <div class="header">
      <!-- código do menu do cabeçalho -->
    </div>
```

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

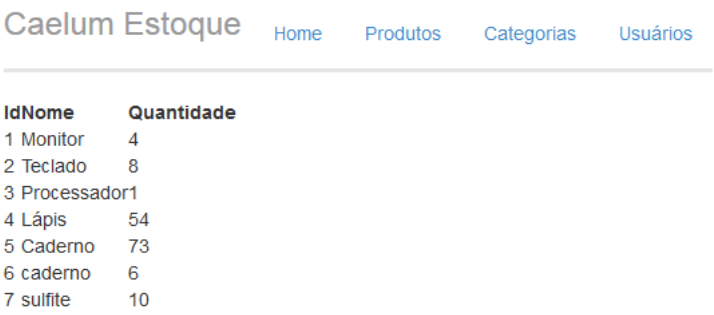
Os links do menu ficarão dentro de uma lista do html, tag `ul`, com as classes `nav nav-pills pull-right`. Além disso, mostraremos também o nome da aplicação dentro de uma tag `h3`:

```
<div class="header">
  <ul class="nav nav-pills pull-right">
    <li>@Html.ActionLink("Home", "Index", "Home")</li>
    <li>@Html.ActionLink("Produtos", "Index", "Produto")</li>
    <li>@Html.ActionLink("Categorias", "Index", "Categoria")</li>
    <li>@Html.ActionLink("Usuários", "Index", "Usuario")</li>
  </ul>
  <h3 class="text-muted">Caelum Estoque</h3>
</div>
```

Depois de colocarmos o código do cabeçalho da aplicação, vamos colocar mais uma `div` que conterá o código da tabela de produtos:

```
<div class="container">
  <div class="header">
    <!-- código do menu -->
  </div>
  <div id="conteudo">
    <!-- Aqui fica o código da tabela de produtos -->
  </div>
</div>
```

Se executarmos a aplicação, veremos a seguinte página:



Veja que a tabela padrão do html não ficou muito boa nesse layout, então utilizaremos uma tabela definida no bootstrap. Dentro da tag table, coloque as classes `table table-hover`, com isso teremos a tabela abaixo:

Com isso terminamos o layout da lista de produtos, agora precisamos replicar esse layout para as outras páginas da aplicação. O menu superior e a importação dos estilos do bootstrap serão utilizados em todas as páginas da aplicação, a única parte diferente entre as páginas é o conteúdo da `div` que contém a tabela de produtos, então teremos que replicar esse código em todas as views da aplicação.

Mas o que aconteceria se precisássemos mudar o layout da aplicação, como copiamos o código para todas as páginas, teríamos que mudar o código de toda aplicação. Para resolvermos esse problema, podemos utilizar os **layout pages** do Asp.Net MVC.

O layout page é um arquivo `cshtml` comum onde colocamos o código que é comum às views da aplicação. Esse arquivo é geralmente colocado dentro de uma pasta chamada `Views/Shared` e tem um nome começado com o símbolo `_` . Então, vamos criar uma nova pasta chamada `Shared` dentro da pasta views do projeto e dentro dessa pasta, colocaremos o arquivo `_Layout.cshtml` com o código que será repetido nas views:

```
<html>
<head>
  <link rel="stylesheet" href="~/Content/Css/bootstrap.css"/>
  <link rel="stylesheet" href="~/Content/Css/Site.css"/>
</head>
<body>
  <div class="container">
    <div class="header">
      <ul class="nav nav-pills pull-right">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("Produtos", "Index", "Produto")</li>
        <li>@Html.ActionLink("Categorias", "Index", "Categoria")</li>
        <li>@Html.ActionLink("Usuários", "Index", "Usuario")</li>
      </ul>
      <h3 class="muted-text">Caelum Estoque</h3>
    </div>
    <div id="conteudo">
      <!-- Aqui fica o conteúdo da view -->
    </div>
  </div>
</body>
</html>
```

No ponto em que queremos colocar o conteúdo da view, precisamos chamar o método `@RenderBody()` do Razor:

```
<html>
<head>
  <link rel="stylesheet" href="~/Content/Css/bootstrap.css"/>
  <link rel="stylesheet" href="~/Content/Css/Site.css"/>
</head>
<body>
  <div class="container">
    <div class="header">
      <ul class="nav nav-pills pull-right">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("Produtos", "Index", "Produto")</li>
        <li>@Html.ActionLink("Categorias", "Index", "Categoria")</li>
        <li>@Html.ActionLink("Usuários", "Index", "Usuario")</li>
      </ul>
      <h3 class="muted-text">Caelum Estoque</h3>
    </div>
    <div id="conteudo">
      @RenderBody()
    </div>
  </div>
</body>
</html>
```

Agora precisamos fazer com que a lista de produtos utilize o layout que acabamos de criar. Para fazer isso, precisamos abrir um bloco de código na view e dentro desse bloco definiremos uma variável chamada `Layout` com o caminho completo para o layout page.

```
@{
  Layout = "caminho completo para o layout page";
}
```

Na string do caminho para a layout page, podemos utilizar novamente o símbolo `~` para referenciarmos a pasta inicial do projeto:

```
@{
  Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Agora podemos apagar todo o código de layout que colocamos inicialmente na view da lista de produtos. Agora só precisamos definir os layouts para as outras views da aplicação, então teremos que, novamente, replicar o bloco que define a variável layout em todas as views. O que pode causar problemas caso tenhamos que, por exemplo, mudar o nome do arquivo de layout.

Para resolvermos esse problema, precisamos definir essa variável `Layout` para todas as views da aplicação, para isso, podemos criar no projeto um novo arquivo especial do razor chamado `_ViewStart.cshtml` dentro da pasta `Views` da aplicação. O arquivo `_ViewStart.cshtml` é o primeiro `cshtml` que é executado quando o razor gera o html e é geralmente utilizado para definir variáveis globais para as views da aplicação.

Então vamos utilizar o `_ViewStart.cshtml` para definir o layout page padrão da aplicação. Dentro da pasta `Views`, crie um novo arquivo chamado `_ViewStart.cshtml` com o código que define a variável `Layout`:

```
@{
  Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Melhorando o aparência dos formulários

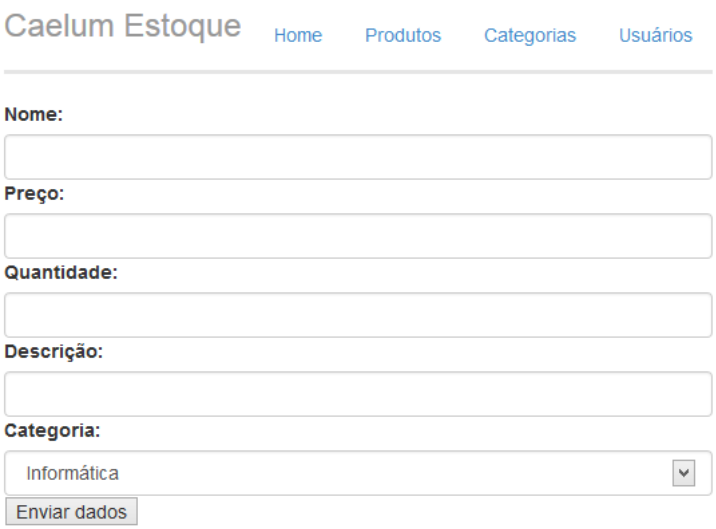
Agora vamos melhorar também a aparêcia dos formulários que criamos na aplicação, para isso, dentro de todas as tags input, colocaremos a classe `form-control` definida dentro do twitter bootstrap:

```
<input name="produto.Nome" value="@ViewBag.Produto.Nome" class="form-control"/>
```

Além disso, colocaremos a classe `form-control` também dentro da tag `select` do cadastro de produtos:

```
<select name="produto.CategoriaId" class="form-control">
  <!-- código das options -->
</select>
```

Após essas modificações, o formulário fica da seguinte forma:



Com isso conseguimos customizar nossa aplicação Asp.Net MVC 5!