02

Conhecendo a View

Transcrição

Como vimos anteriormente, temos a aplicação rodando no browser, e ela já vem com uma página de modelo. Isso porque o Visual Studio criou a aplicação ASP.NET Core MVC a partir de um template, que contém uma série de conteúdos, como imagens, e outras informações que são úteis para quem está começando a explorar o ASP.NET Core.

Mas o que é o ASP.NET Core? Trata-se de um framework, criado pela Microsoft, a partir do ASP.NET clássico, também chamado de ASP.NET 4.X. A empresa reescreveu esta versão clássica, porém, permitiu que o novo pudesse rodar em outros sistemas operacionais, como Linux e Mac OSx.

Sendo assim, você pode utilizar o projeto ora disponibilizado em outros servidores, e eles serão executados normalmente — algo que não era permitido na versão clássica. Portanto, além de ser multiplataforma, a nova versão é mais modular. Podemos baixar, individualmente, todos os componentes que fazem parte do ASP.NET Core diretamente do gerenciador de pacotes.

Retornaremos ao Visual Studio para descobrirmos a origem da página inicial da aplicação, que estamos observando no navegador. No projeto `CasaDoCodigo` , temos os componentes que fazem parte da aplicação. Entretanto, não há um arquivo único responsável por fazer o carregamento da página HTML, não temos um arquivo de página. O que temos, na verdade, é a visualização que vimos anteriormente. À ela é dado o nome de *view*, e ele é um dos componentes do padrão MVC.

No projeto, temos uma pasta chamada "Views", na qual há uma subpasta denominada "Home". Nela, estão agrupadas outras *views*, ou visualizações, que fazem parte da aplicação. Então, qual é a origem da nossa página inicial? Ela é carregada, por padrão, a partir do arquivo `Index.cshtml` . Abriremos este arquivo. Veremos o código HTML utilizado para gerar nossa página no browser. Analisaremos a extensão `.cshtml` — indicativo de que o arquivo é uma mistura de C# com HTML — com mais detalhes em uma oportunidade futura.

Por enquanto, faremos somente uma modificação neste arquivo, removendo uma expressão em Inglês, "Learn how to build ASP.NET apps that can run anywhere.", e trocaremos pela frase em Português, "Aprenda a construir apps ASP.NET que rodam em todo lugar". Rodando a aplicação novamente, veremos que a mensagem foi alterada no browser.

```
@{
    ViewData["Title"] = "Home Page";
}

<div id="myCarousel" class="carousel slide" data-ride="carousel" data-interval="6000">
    <ol class="carousel-indicators">
        <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
        <li data-target="#myCarousel" data-slide-to="1"></li>
        <li data-target="#myCarousel" data-slide-to="2"></li>
        <li data-target="#myCarousel" data-slide-to="3"></li>
    </ol>
    <div class="carousel-inner" role="listbox">
        <div class="item active">
            <img src="~/images/banner1.svg" alt="ASP.NET" class="img-responsive" />
            <div class="carousel-caption" role="option">
                <p>
                    Aprenda a construir apps ASP.NET que rodam em todo lugar.
                    <a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkID=52502‘
```

```
                    Learn More
                </a>
            </p>
        </div>
    </div>
    <div class="item">
        <img src="~/images/banner2.svg" alt="Visual Studio" class="img-responsive" />
        <div class="carousel-caption" role="option">
            <p>
                There are powerful new features in Visual Studio for building modern web apps.
                <a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkID=525030&
                    Learn More
                </a>
            </p>
        </div>
    </div>
    <div class="item">
        <img src="~/images/banner3.svg" alt="Package Management" class="img-responsive" />
        <div class="carousel-caption" role="option">
            <p>
                Bring in libraries from NuGet and npm, and automate tasks using Grunt or Gulp.
                <a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkID=525029&
                    Learn More
                </a>
            </p>
        </div>
    </div>
    <div class="item">
        <img src="~/images/banner4.svg" alt="Microsoft Azure" class="img-responsive" />
        <div class="carousel-caption" role="option">
            <p>
                Learn how Microsoft's Azure cloud platform allows you to build, deploy, and scal
                <a class="btn btn-default" href="https://go.microsoft.com/fwlink/?LinkID=525027&
                    Learn More
                </a>
            </p>
        </div>
    </div>
</div>
<a class="left carousel-control" href="#myCarousel" role="button" data-slide="prev">
    <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
</a>
<a class="right carousel-control" href="#myCarousel" role="button" data-slide="next">
    <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
</a>
</div>
```

Assim, vimos como mudar um arquivo no servidor da nossa aplicação, para que isso seja refletido no browser, ou seja, como uma página é carregada a partir de uma view. O HTML é formado a partir de uma view do MVC, e é carregado para o browser. O que mais, além da view, compõe uma página de um padrão MVC? Temos também um outro componente, ao qual damos o nome de modelo, ou *model*. Ele nos ajuda a gerar uma página dinâmica.

Imaginemos que temos um relatório e, a cada vez que o abrimos, temos dados diferentes. Isso acontece porque as informações são diferentes e fornecidas por componentes chamados de modelos. Além disso, temos o *controller*. Ele reúne as informações da view e do modelo, para compor uma página HTML pronta para o browser. No navegador, temos um endereço para acessar a página da home, que em nosso caso é `localhost:58246`, por que conseguimos acessar a partir do `localhost` mais o número da porta?

Porque, por padrão, caímos sempre na view chamada `Index.cshtml`, que está dentro da pasta "Home". Contudo, poderíamos inserir no endereço também o `localhost:58246/Home/Index`, e isso nos trará justamente a visualização da página inicial.

Adiante, veremos como acessar as outras views da nossa aplicação.