06

Configurando session, gravando e lendo PedidoId na sessão

Transcrição

Estamos passando o código do produto na rota para adicioná-lo ao carrinho, porém, note que a navegação passa a informação de uma página a outra. De que forma a aplicação saberá que estamos utilizando o mesmo pedido?

É importante perceber que, por definição, a web não possui estado, ou seja, é chamada de ***stateless***. Significa que cada requisição feita a uma página diferente nos traz uma nova história, e que não estamos levando a informação do pedido como deveríamos. Então, o que fazemos para manter um estado entre as páginas, para que a aplicação saiba que não importa em qual página esteja, estamos trabalhando com o mesmo pedido?

Utilizaremos uma técnica chamada **Sessão**, que mantém o estado de maneira artificial e controlada pelo servidor — estudaremos como configurá-la para mantermos o `id` do pedido ao longo da nossa navegação. Vamos salvar a aplicação e entrar na classe responsável pela sua configuração, `Startup`. Nela, encontraremos dois métodos:

- `ConfigureServices()`, que é onde adicionamos o serviço à aplicação;
- `Configure()`, em que definimos que iremos realmente utilizar o determinado serviço.

O primeiro é usado para adição, e o segundo, para configurar a sua utilização.

Em `ConfigureServices()`, adicionaremos o serviço de sessão `AddSession()`, logo após o método `AddMvc()`. Para fazê-lo, precisaremos incluir também outro serviço, responsável por manter as informações na memória, conforme vamos navegando, o serviço de *cache*, mais especificamente, o ***cache distribuído em memória***, por meio de `AddDistributedMemoryCache()`:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddDistributedMemoryCache();
    services.AddSession();
        services.TryAddSingleton<IHttpContextAccessor, HttpContextAccessor>();



    // código omitido
}
```

Agora que configuramos o serviço neste método, vamos utilizá-lo no outro, `Configure()`, que expandiremos, e incluiremos:

```
app.UseStaticFiles();
app.UseSession();
app.UseMvc(routes =>
{
    // código omitido
});
```

Com isso, configuramos a sessão a ser usada; entraremos no repositório de pedido e criaremos um método para obtermos o `id` do pedido que será armazenado na sessão. Em `PedidoRepository.cs`, portanto, criaremos um método privado com tipo de retorno `int?`, com ponto de interrogação (`?`) para indicar que ele pode ser nulo, seguido de `GetPedidoId()`.

Começaremos acessando o objeto da sessão para chegarmos ao `id` do pedido que será armazenado, fornecido por um componente chamado de ***HTTP Context Accessor*** e, para declará-lo, poderemos criar um campo privado na classe `PedidoRepository` de nome `contextAccessor`.

```
public class PedidoRepository : BaseRepository<Pedido>, IPedidoRepository
{
    private readonly IHttpContextAccessor contextAccessor;

    public PedidoRepository(ApplicationContext contexto) : base(contexto)
    {
    }

    private int? GetPedidoId()
    {

    }
}
```

Em seguida, faremos a classe receber este objeto através da injeção de dependência que, por padrão, já está registrado, bastando colocá-lo como parâmetro do nosso construtor, com o qual, mais abaixo no código, faremos a atribuição do campo local. Para obtermos o pedido de `id` por meio de `contextAccessor`, colocaremos sua referência e `HttpContext`, após o qual acessaremos a sessão.

Então, obteremos o valor que será `int`, indicado por `GetInt32()`. Entre parênteses, incluiremos o nome da chave do dado que armazenaremos e, assim que o obtivermos, teremos que retorná-lo, então usaremos `return`. Para gravarmos `pedidoId` na sessão, criaremos outro método privado, no qual colocaremos o mesmo objeto `contextAccessor`:

```
public PedidoRepository(ApplicationContext contexto,
    IHttpContextAccessor contextAccessor) : base(contexto)
{
    this.contextAccessor = contextAccessor;
}

private int? GetPedidoId()
{
    return contextAccessor.HttpContext.Session.GetInt32("pedidoId");
}

private void SetPedidoId(int pedidoId)
{
    contextAccessor.HttpContext.Session.SetInt32("pedidoId", pedidoId);
}
```

Com isso, definimos os dois métodos, para ler e gravar o pedido de `id` na sessão da nossa aplicação ASP.NET Core MVC.