09

Para saber mais: Como Ler Configurações no Controller?

Para Saber Mais...

Como Ler Configurações no Controller?

Você pode ter se deparado com esse problema da vida real, fora do contexto do nosso curso: **como ler configurações do ASP.NET Core 2.0 a partir de um controller?** Vamos partir do pressuposto de que você tenha uma aplicação de blog com uma estrutura de configuração definida no arquivo `appsetings.json`.

Vamos começar pelo arquivo `appsetings.json`. Digamos que sua configuração seja definida pelo arquivo abaixo:

```
{
    "ConnectionStrings": {
        "Blog": "Server=(localdb)\\mssqllocaldb;Database=MeuBlog;Trusted_Connection=True;"
    },
    "Security": {
        "Language" : "pt-BR",
        "SuperUser": {
            "Login": "bgiorgione",
            "Email": "bgiorgione@bgiorgione.com.br",
            "ShowEmail": "true"
        },
        "Admin": {
            "Login": "moliveira",
            "Email": "mclricardo@gmail.com.br",
            "ShowEmail": "false"
        }
    }
}
```

Listagem: o arquivo appsetings.json

Todos os método de que precisamos precisam vir da classe `Configuration`, que é uma implementação da interface `IConfiguration`.

O primeiro passo **seria** configurar o *mecanismo de injeção de dependência* do ASP.NET Core para obter o objeto de configuração. Mas felizmente esse objeto já vem automaticamente registrado internamente no contêiner de injeção de dependência do ASP.NET Core, portanto você **não precisa** adicionar o código `services.AddSingleton<IConfiguration>(Configuration);` na sua classe `Startup`.

Agora, como vimos no curso, basta adicionar um parâmetro `IConfiguration` no construtor do controller para começarmos a obter as configurações:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;

namespace Blog.Controllers
{
```

```
    public class BlogController : Controller
    {
        private IConfiguration _configuration;

        public BlogController(IConfiguration Configuration)
        {
            _configuration = Configuration;
        }

        public IActionResult Usuarios()

        {
            // Configuração usando a string de chave
            ViewData["SecurityLanguage"] = _configuration["Security:Language"];

            // Configuração usando método GetSection

            ViewData["SecuritySuperUserLogin"] =
            _configuration.GetSection("Security").GetSection("SuperUser").GetSection("Login");

            // Configuração usando GetSection e string de chave ao mesmo tempo
            ViewData["SecuritySuperUserEmail"] =
                _configuration.GetSection("Security")["SuperUser:Email"];

            // Configuração usando GetSection e string de chave ao mesmo tempo
            ViewData["SecuritySuperUserShowEmail"] =
                _configuration.GetSection("Security")["SuperUser:ShowEmail"];

            return View();
        }
    }
}
```

Listagem: o arquivo BlogController.cs

No exemplo acima, populamos o `ViewData` com as configuração. Mas você pode utilizar as configurações como quiser, claro.

Agora basta exibir os dados na view `Usuarios`:

```
<p>
    SecurityLanguage <strong>@ViewData["SecurityLanguage"]</strong><br />
    SecuritySuperUserLogin <strong>@ViewData["SecuritySuperUserLogin"]</strong><br />
    SecuritySuperUserEmail <strong>@ViewData["SecuritySuperUserEmail"]</strong>
    SecuritySuperUserShowEmail <strong>@ViewData["SecuritySuperUserShowEmail"]</strong>
</p>
```

Listagem: o arquivo Usuarios.cshtml