03

Configurando Banco de Dados

Transcrição

Nesta aula, criaremos um banco de dados vazio para podermos fazer a migração das classes e gerar as tabelas do banco de dados. Clicaremos no menu "View" do Visual Studio, na barra de menu superior, e selecionaremos a opção "SQL Server Object Explorer". Teremos uma estrutura contendo a pasta "Databases", clicaremos sobre ela com o botão direito do mouse e, em seguida, sobre "Add New Database".

Daremos a ele o nome de `CasaDoCodigo` e, assim, teremos nosso banco de dados vazio. Precisamos passar as informações do banco para a nossa aplicação, por exemplo, qual o servidor e o seu nome. Por enquanto, no banco `CasaDoCodigo` temos somente as tabelas do sistema, para criarmos nossas próprias, precisaremos da string de conexão deste.

Clicaremos sobre o nome do banco e pressionaremos a tecla "F4" para abrirmos a janela de propriedades. Em *Connection string*, temos o texto contendo a string de conexão. Copiaremos este texto. No projeto "CasaDoCodigo", temos uma maneira padrão do ASP.NET Core de definir as configurações de onde acessar este banco de dados.

Abriremos o arquivo `appsettings.json`, na pasta do projeto "CasaDoCodigo". Nele, há somente uma configuração de log. Portanto, onde será criada a configuração de banco de dados? Teremos uma nova entrada, que chamaremos de `ConnectionStrings`:

```
{
    "Logging": {
        "IncludeScopes": false,
        "LogLevel": {
            "Default": "Warning"
        }
    },
    "ConnectionStrings": {

    }
}
```

Inseriremos a conexão default, que terá este nome e, por padrão, faremos a conexão no banco de dados que acabamos de criar:

```
{
    "Logging": {
        "IncludeScopes": false,
        "LogLevel": {
            "Default": "Warning"
        }
    },
    "ConnectionStrings": {
        "Default": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=CasaDoCodigo;Integrated Secu
    }
}
```

Com isso, o ASP.NET Core terá informações necessárias para poder acessar o banco. Retornaremos à aplicação, para definirmos como utilizar a `ConnectionString`. No projeto, temos uma classe de configuração chamada `Startup.cs`. Nela, há dois métodos principais, o `ConfigureServices()` e o `Configure()`. Abriremos o `ConfigureServices()` e definiremos um método de configuração para o nosso banco de dados.

Temos um serviço já existente, que é o MVC. Para adicionarmos um novo, chamaremos `services` com o método `AddDbContext`, passando o nome da classe do contexto do banco de dados, no caso, o `ApplicationContext()`.

Faremos a chamada para este último método passando as opções de configuração do contexto como argumento, ou seja, a definição da utilização da `ConnectionString`. Assim, passaremos o nome do parâmetro, seguido pelo operador lâmbda `(options => options)`.

A seguir, chamaremos o parâmetro `options` com o método que nos permite utilizar, como nosso banco de dados, o SQL Server. Precisamos importar o `namespace` correspondente, utilizando o atalho "Ctrl + ." sobre `UseSqlServer`. Por fim, passaremos como parâmetro a `ConnectionString`.

```
namespace CasaDoCodigo
{
    public class Startup
    {
        public Startup(IConfiguration configuration)

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();

            services.AddDbContext<ApplicationContext>(options =>
                options.UseSqlServer(connectionString)
            );
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pip
        public void Configure(IApplicationBuilder app, IHostingEnvironment env,
            IServiceProvider serviceProvider)
    }
}
```

Feito isso, a `ConnectionString` ainda não foi reconhecida. Na realidade, esta variável sequer existe em nosso código. Por isso, precisamos criá-la, buscando as configurações presentes no arquivo `AppSettings.json`. Acessaremos o objeto `Configuration` e o método `GetConnectionString()`, passando o nome da configuração, que no nosso caso é `Default`:

```
namespace CasaDoCodigo
{
    public class Startup
    {
        public Startup(IConfiguration configuration)

        public IConfiguration Configuration { get; }
```

```
        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();

            string connectionString = Configuration.GetConnectionString("Default");

            services.AddDbContext<ApplicationContext>(options =>
                options.UseSqlServer(connectionString)
            );
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request pip
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    }
}
```

Assim, geramos o banco de dados a partir das classes do nosso modelo, com base nesta configuração.