

Propriedades Shadow e indexer

09/10/2020 • 3 minutos para o fim da leitura •  

Neste artigo

[Propriedades da sombra da chave estrangeira](#)

[Configurando propriedades de sombra](#)

[Acessando Propriedades de sombra](#)

[Configurando propriedades do indexador](#)

[Tipos de entidade do recipiente de propriedades](#)

Propriedades de sombra são propriedades que não são definidas em sua classe de entidade do .NET, mas são definidas para esse tipo de entidade no modelo de EF Core. O valor e o estado dessas propriedades são mantidos puramente no controlador de alterações. As propriedades de sombra são úteis quando há dados no banco que não devem ser expostos nos tipos de entidade mapeada.

As propriedades do indexador são propriedades de tipo de entidade, que são apoiadas por um [indexador](#) na classe de entidade do .net. Eles podem ser acessados usando o indexador nas instâncias de classe do .NET. Ele também permite que você adicione propriedades adicionais ao tipo de entidade sem alterar a classe CLR.

Propriedades da sombra da chave estrangeira

As propriedades de sombra são usadas com mais frequência para propriedades de chave estrangeira, em que a relação entre duas entidades é representada por um valor de chave estrangeira no banco de dados, mas a relação é gerenciada nos tipos de entidade usando propriedades de navegação entre os tipos de entidade. Por convenção, o EF introduzirá uma propriedade de sombra quando uma relação for descoberta, mas nenhuma propriedade de chave estrangeira for encontrada na classe de entidade dependente.

A propriedade será nomeada <navigation property name><principal key property name> (a navegação na entidade dependente, que aponta para a entidade principal, é usada para a nomenclatura). Se o nome da propriedade da chave principal incluir o nome da propriedade de navegação, o nome será apenas <principal key property name> . Se não houver nenhuma propriedade de navegação na entidade dependente, o nome do tipo de entidade de segurança será usado em seu lugar.

Por exemplo, a listagem de código a seguir resultará em uma `BlogId` propriedade de sombra introduzida na `Post` entidade:

C#	 Copiar
<pre>internal class MyContext : DbContext { public DbSet<Blog> Blogs { get; set; } public DbSet<Post> Posts { get; set; } } public class Blog { public int BlogId { get; set; } public string Url { get; set; } public List<Post> Posts { get; set; } } public class Post { public int PostId { get; set; } public string Title { get; set; } public string Content { get; set; } // Since there is no CLR property which holds the foreign // key for this relationship, a shadow property is created. public Blog Blog { get; set; } }</pre>	

Configurando propriedades de sombra

Você pode usar a API Fluent para configurar propriedades de sombra. Depois de ter chamado a sobrecarga de cadeia de caracteres do `Property`, você pode encadear qualquer uma das chamadas de configuração para outras propriedades. No exemplo a seguir, como `Blog` não tem nenhuma propriedade CLR denominada `LastUpdated`, uma propriedade Shadow é criada:


C#	 Copiar
<pre>internal class MyContext : DbContext { public DbSet<Blog> Blogs { get; set; } protected override void OnModelCreating(ModelBuilder modelBuilder) { modelBuilder.Entity<Blog>() .Property<DateTime>("LastUpdated"); } }</pre>	

```
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
}
```


Se o nome fornecido ao `Property` método corresponder ao nome de uma propriedade existente (uma propriedade de sombra ou uma definida na classe de entidade), o código configurará essa propriedade existente em vez de introduzir uma nova propriedade de sombra.

Acessando Propriedades de sombra

Os valores de propriedade de sombra podem ser obtidos e alterados por meio da `ChangeTracker` API:

C#	 Copiar
<pre>context.Entry(myBlog).Property("LastUpdated").CurrentValue = DateTime.Now;</pre>	


As propriedades de sombra podem ser referenciadas em consultas LINQ por meio do `EF.Property` método estático:

C#	 Copiar
<pre>var blogs = context.Blogs .OrderBy(b => EF.Property<DateTime>(b, "LastUpdated"));</pre>	

As propriedades de sombra não podem ser acessadas após uma consulta sem rastreamento, pois as entidades retornadas não são rastreadas pelo rastreador de alterações.

Configurando propriedades do indexador

Você pode usar a API Fluent para configurar as propriedades do indexador. Depois de chamar o método `IndexerProperty`, você poderá encadear qualquer uma das chamadas de configuração para outras propriedades. No exemplo a seguir, `Blog` tem um indexador definido e ele será usado para criar uma propriedade do indexador.

C#	 Copiar
<pre>protected override void OnModelCreating(ModelBuilder modelBuilder) {</pre>	

```
modelBuilder.Entity<Blog>().IndexerProperty<DateTime>("LastUpdated");  
}
```

Se o nome fornecido ao `IndexerProperty` método corresponder ao nome de uma propriedade do indexador existente, o código configurará essa propriedade existente. Se o tipo de entidade tiver uma propriedade, que é apoiada por uma propriedade na classe de entidade, uma exceção será gerada, pois as propriedades do indexador só devem ser acessadas por meio do indexador.

Tipos de entidade do recipiente de propriedades

ⓘ Observação

O suporte para tipos de entidade de recipiente de propriedades foi introduzido no EF Core 5,0.

Os tipos de entidade que contêm apenas propriedades do indexador são conhecidos como tipos de entidade de conjunto de propriedades. Esses tipos de entidade não têm propriedades de sombra, em vez disso, o EF criará Propriedades do indexador. Atualmente, só tem `Dictionary<string, object>` suporte como um tipo de entidade de conjunto de propriedades. Ele deve ser configurado como um tipo de entidade compartilhada com um nome exclusivo e a `DbSet` propriedade correspondente deve ser implementada usando uma `Set` chamada.

C#

 Copiar

```
internal class MyContext : DbContext  
{  
    public DbSet<Dictionary<string, object>> Blogs => Set<Dictionary<string,  
object>>("Blog");  
  
    protected override void OnModelCreating(ModelBuilder modelBuilder)  
    {  
        modelBuilder.SharedTypeEntity<Dictionary<string, object>>(  
            "Blog", bb =>  
            {  
                bb.Property<int>("BlogId");  
                bb.Property<string>("Url");  
                bb.Property<DateTime>("LastUpdated");  
            });  
    }  
}
```

Esta página é útil?

 Yes  No
