

第08期-实体关系配置

2019年4月16日 21:03

显式外键配置

```
referenceCollectionBuilder.HasForeignKey(p => p.BlogForeignKey);

modelBuilder.Entity<Car>().HasKey(c => new { c.State, c.LicensePlate });

referenceCollectionBuilder.HassForeignKey(s => new { s.CarState, s.CarLicensePlate })

referenceCollectionBuilder.HasForeignKey("BlogId");
```

显式主体键

如果你想要引用主体实体主键之外的属性作为外键，可通过 HasPrincipalKey 配置。

```
modelBuilder.Entity<Post>().HasOne(p => p.Blog).WithMany(b => b.Posts).HasForeignKey(p
=> p.BlogUrl).HasPrincipalKey(b => b.Url);

modelBuilder.Entity<Post>().HasOne(p => p.Blog).WithMany(b => b.Posts).HasForeignKey(p
=> new { p.BlogUrl, p.BlogState }).HasPrincipalKey(b => new { b.Url, b.State });
```

必需和可选的关系

可使用 Fluent API 来配置关系为必需或可选，使用隐藏属性时非常有用。

```
modelBuilder.Entity<Post>().HasOne(p => p.Blog).WithMany(b => b.Posts).IsRequired();
```

一对一关系

一对一关系两端都是引用导航属性，无法判断那个作为主体实体，推荐显式指定外键属性。

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

    public BlogImage BlogImage { get; set; }
}
```

```

public class BlogImage
{
    public int BlogImageId { get; set; }
    public byte[] Image { get; set; }
    public string Caption { get; set; }

    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}

```

如果使用 Fluent API 配置此关系，则使用 HasOne 和 WithOne 方法。

```

modelBuilder.Entity<Blog>().HasOne(p => p.BlogImage).WithOne(i =>
i.Blog).HasForeignKey<BlogImage>(b => b.BlogForeignKey);

```

多对多关系

关系型数据库中不支持多对多的映射，通过建立中间表连接，使用一对多的方式模拟多对多关系。例如：用户角色表，学生选课表。

```

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public List<PostTag> PostTags { get; set; }
}

```

```

public class Tag
{
    public string TagId { get; set; }

    public List<PostTag> PostTags { get; set; }
}

```

```

public class PostTag
{
    public int PostId { get; set; }
    public Post Post { get; set; }

    public string TagId { get; set; }
    public Tag Tag { get; set; }
}

```

```

}

class MyContext : DbContext
{
    public DbSet<Post> Posts { get; set; }
    public DbSet<Tag> Tags { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<PostTag>()
            .HasKey(t => new { t.PostId, t.TagId });

        modelBuilder.Entity<PostTag>()
            .HasOne(pt => pt.Post)
            .WithMany(p => p.PostTags)
            .HasForeignKey(pt => pt.PostId);

        modelBuilder.Entity<PostTag>()
            .HasOne(pt => pt.Tag)
            .WithMany(t => t.PostTags)
            .HasForeignKey(pt => pt.TagId);
    }
}

```