

第54期-动态模型配置

2019年9月9日 21:57

使用元数据动态配置模型

动态配置表名（默认复数）

```
//This will singularize all table names
foreach (ImmutableEntityType entityType in modelBuilder.Model.GetEntityTypes())
{
    entityType.Relational().TableName = entityType.DisplayName();
}
```

动态修改表配置

```
foreach (ImmutableEntityType entityType in modelBuilder.Model.GetEntityTypes())
{
    modelBuilder.Entity(entityType.Name).ToTable($"T_{entityType.DisplayName()}");
}
```

动态修改列配置

```
foreach (ImmutableEntityType entityType in modelBuilder.Model.GetEntityTypes())
{
    foreach (ImmutableProperty property in entityType.GetProperties().Where(p => p.ClrType ==
typeof(string)))
    {
        property.SetMaxLength(20);
    }
}
```

```
foreach (ImmutableEntityType entityType in modelBuilder.Model.GetEntityTypes())
{
    foreach (ImmutableProperty property in entityType.GetProperties().Where(p => p.ClrType ==
typeof(string)))
    {
        modelBuilder.Entity(entityType.Name).Property(property.Name).HasColumnName($"C_{pr
operty.Name}");
    }
}
```

配置上下文

```
virtual void OnConfiguring(DbContextOptionsBuilder optionsBuilder);
```

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {

```

```

        optionsBuilder.UseSqlServer("connectionString");
    }

    base.OnConfiguring(optionsBuilder);
}

```

配置模型

```

virtual void OnModelCreating(ModelBuilder modelBuilder);

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Blog>().Property(b => b.Url).HasColumnName("BlogUrl");

    base.OnModelCreating(modelBuilder);
}

```

最常规最简单的写法

```

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Blog>().Property(b => b.Url).HasColumnName("BlogUrl");
    modelBuilder.Entity<Blog>().Property(b => b.Name).HasMaxLength(50);

    modelBuilder.Entity<Post>().Property(b => b.Title).HasMaxLength(30);
    modelBuilder.Entity<Post>().Property(b => b.Content).HasMaxLength(500);

    base.OnModelCreating(modelBuilder);
}

```

使用表达式分离配置的写法

```

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Blog>(buildAction =>
    {
        buildAction.Property(b => b.Url).HasColumnName("BlogUrl");
        buildAction.Property(b => b.Name).HasMaxLength(50);
    });

    modelBuilder.Entity<Post>(buildAction =>
    {
        buildAction.Property(b => b.Title).HasMaxLength(30);
        buildAction.Property(b => b.Content).HasMaxLength(500);
    });

    base.OnModelCreating(modelBuilder);
}

```

使用方法分离配置的写法

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Blog>(ConfigureBlog);
    modelBuilder.Entity<Post>(ConfigurePost);

    base.OnModelCreating(modelBuilder);
}
```

```
private void ConfigureBlog(EntityTypeBuilder<Blog> buildAction)
{
    buildAction.Property(b => b.Url).HasColumnName("BlogUrl");
    buildAction.Property(b => b.Name).HasMaxLength(50);
}
```

```
private void ConfigurePost(EntityTypeBuilder<Post> buildAction)
{
    buildAction.Property(b => b.Title).HasMaxLength(30);
    buildAction.Property(b => b.Content).HasMaxLength(500);
}
```

使用类分离配置的写法

```
public class BlogConfiguration : IEntityTypeConfiguration<Blog>
{
    public void Configure(EntityTypeBuilder<Blog> builder)
    {
        builder.Property(b => b.Url).HasColumnName("BlogUrl");
        builder.Property(b => b.Name).HasMaxLength(50);
    }
}
```

```
public class PostConfiguration : IEntityTypeConfiguration<Post>
{
    public void Configure(EntityTypeBuilder<Post> builder)
    {
        builder.Property(b => b.Title).HasMaxLength(30);
        builder.Property(b => b.Content).HasMaxLength(500);
    }
}
```

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.ApplyConfiguration(new BlogConfiguration());
    modelBuilder.ApplyConfiguration(new PostConfiguration());

    base.OnModelCreating(modelBuilder);
}
```

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());
}
```

```
        base.OnModelCreating(modelBuilder);  
    }
```

重写 DbContext 相关的方法

DbContext 提供了诸多的虚方法，可在子类中直接复写对应的方法，在方法执行前后自定义操作。

```
public override int SaveChanges()  
{  
    //TODO  
    return base.SaveChanges();  
}
```

```
public override EntityEntry Remove(object entity)  
{  
    //TODO  
    return base.Remove(entity);  
}
```