将类型包括到模型中意味着，EF 会有该类型的元数据，并且会尝试从数据库读取实例，以及将实例写入到数据库中。

## **约定**

在上下文的 DbSet 属性中公开的类型会包括在模型中。

在 OnModelCreating 方法中提及的类型也将包括在其中。

通过以递归方式浏览已发现类型的导航属性而找到的任何类型也会包括在模型中。

****例如，以下代码列表中发现所有三种类型：****

Blog，因为它是在上下文的 DbSet 属性中公开的

Post，因为它是通过 Blog.Posts 导航属性发现的

AuditEntry，因为它在 OnModelCreating 中提及

class MyContext : DbContext

{

public DbSet<Blog> Blogs { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<AuditEntry>();

}

}

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; }

public Blog Blog { get; set; }

}

public class AuditEntry

{

public int AuditEntryId { get; set; }

public string Username { get; set; }

public string Action { get; set; }

}

## **数据注释**

可以使用数据注释来从模型中排除类型。

public class Blog

{

public int BlogId { get; set; }

public string Url { get; set; }

public BlogMetadata Metadata { get; set; }

}

[NotMapped]

public class BlogMetadata

{

public DateTime LoadedFromDatabase { get; set; }

}

## **Fluent API**

Fluent API 可用于从模型中排除类型。

class MyContext : DbContext

{

public DbSet<Blog> Blogs { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Ignore<BlogMetadata>();

}

}

public class Blog

{

public int BlogId { get; set; }

public string Url { get; set; }

public BlogMetadata Metadata { get; set; }

}

public class BlogMetadata

{

public DateTime LoadedFromDatabase { get; set; }

}