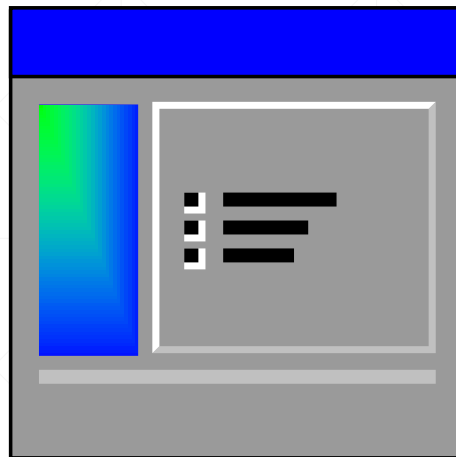
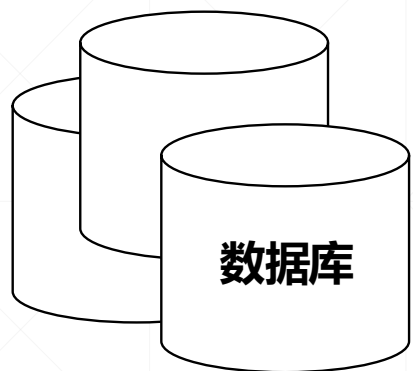


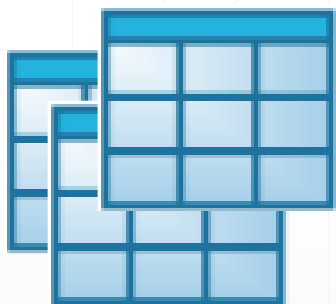
Entity Framework 概述

北京理工大学计算机学院
金旭亮

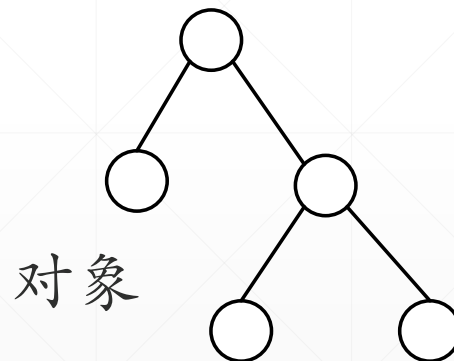
1 Entity Framework与ORM



应用程序



表



对象

在实际开发中，人们设计了许多“**对象/关系映射 (ORM: Object Relational Mapping)**”框架，使应用程序可以使用一种“纯”的对象模型来访问关系数据库中的数据。

当前主流编程语言或开发平台，大都提供了可用的ORM软件。

List of object-relational mapping software

From Wikipedia, the free encyclopedia

This is a list of well-known [object-relational mapping](#) software. It is not up-to-date or all-inclusive.

Contents [\[hide\]](#)

- 1 C++
- 2 Flex
- 3 Java
- 4 JavaScript
- 5 iOS
- 6 .NET
- 7 Object Pascal (Delphi)
- 8 Objective-C, Cocoa
- 9 Perl
- 10 PHP
- 11 Python
- 12 Ruby
- 13 Smalltalk
- 14 Visual Basic 6.0
- 15 See also
- 16 References

http://en.wikipedia.org/wiki/List_of_object-relational_mapping_software

实际开发中的困境.....

单数据表变更

- 新加字段或删除字段
- 原有字段改名、改类型
- 表名修改
- 新建、删除表

多数据表变更

- 关联变更：由一对一转换为一对多或多对多关联
- 表的拆分与合并



大批代码必须修改、重写和测试，之后重新发布和部署.....

ORM是解脱困境的一种尝试和努力

使用ORM，能减少这种数据库结构的变更对现有软件系统的影响应对不断变化的需求和软件运行环境，让软件系统更好地支持重构。

在性能并非主要关心因素，可扩展性及可维护性更为重要的场景下，使用ORM是合理的选择。

ORM使许多日常工作自动化，带来了较高的开发效率，降低了开发成本。

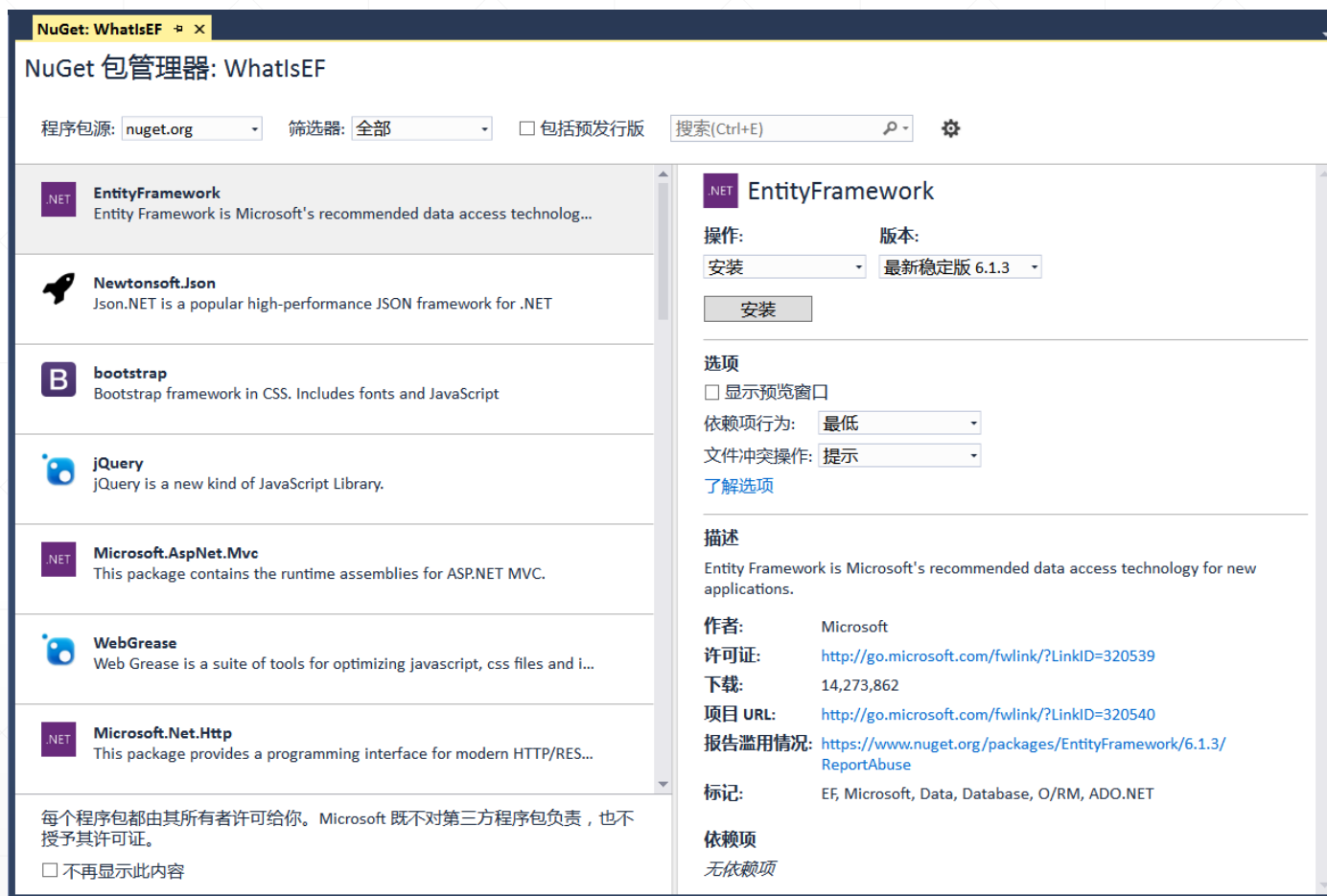
Entity Framework（简称EF）是.NET平台下一个优秀的ORM框架。

为什么学EF？

- **命好**：本身足够优秀，又得到微软官方青睐
- **开源**：不花钱，且有足够的资源投入，持续完善中。
- **强大**：可以访问多种数据库（如MySQL、SQL Server, Oracle、DB2、SQLite、PostgreSQL等）。
- **灵活**：它支持多种开发模式，能胜任多数开发场景。
- **省时省力**：自动化程度相当高，工具出色。

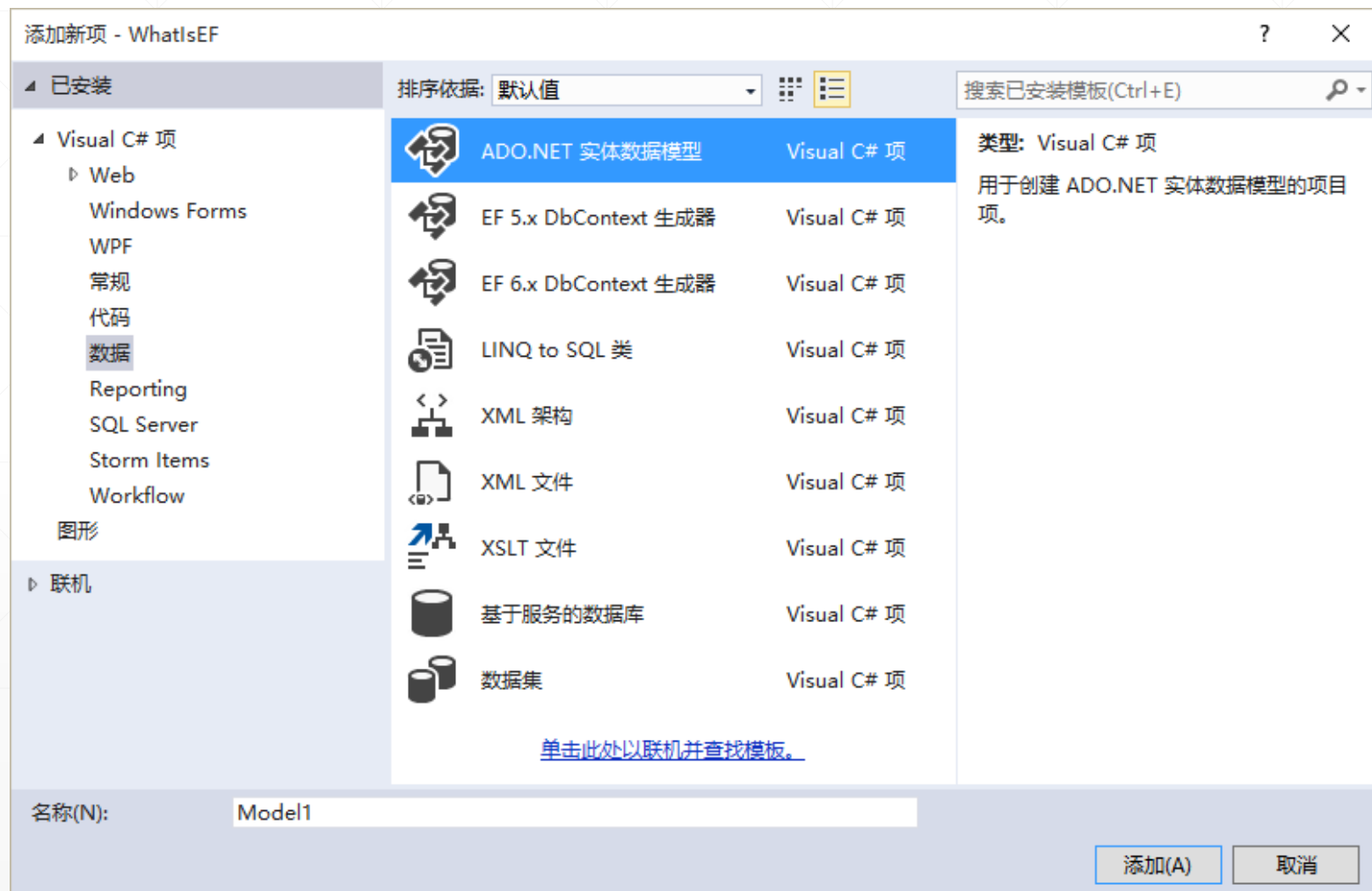
2 在开发中使用Entity Framework

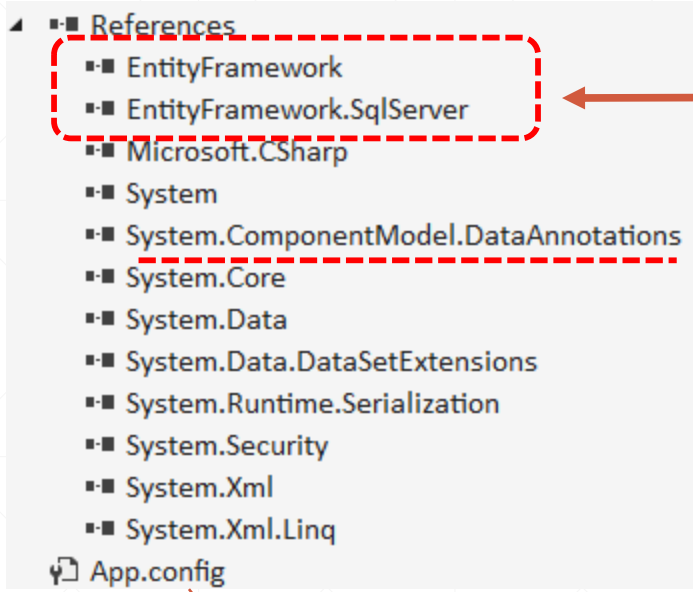
网络可用时，可以使用**NuGet**给项目添加EF支持：



使用这种方法，可以添加或更新到最新版本的EF，也能为任意类型的项目添加EF支持，是最常用的一种给项目添加EF支持的方式。

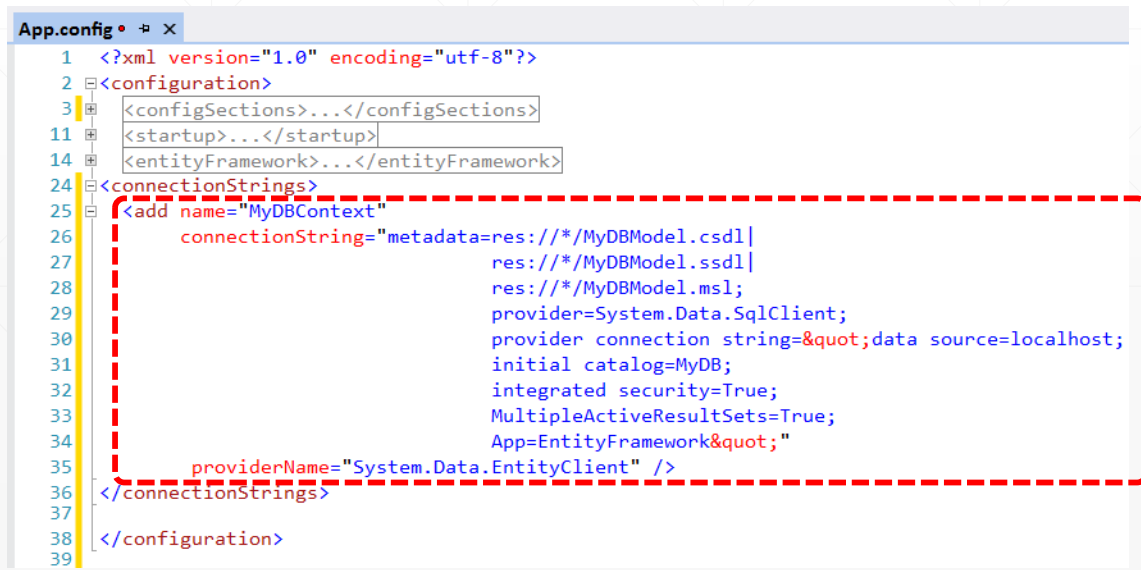
也可以通过给项目添加“**ADO.NET 实体数据模型**”，使用Visual Studio安装时采用的默认版本。





Entity Framework的核心程序集

包容可用于数据实体类的诸多Attribute



EF 从项目的web.config或app.config文件中读取连接数据库所需的各种信息，我们通常在此放置数据库连接字符串

两种类型的数据库连接字符串

```
<connectionStrings>
  <add name="OnlyTestContext"
    connectionString="data source=localhost;
    initial catalog=OnlyTest;
    integrated security=True;
    MultipleActiveResultSets=True;
    App=EntityFramework"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

“代码优先 (Code First)”
开发模式

```
<connectionStrings>
  <add name="NorthwindEntities"
    connectionString="metadata=res://*/NorthwindModel.csdl|
    res://*/NorthwindModel.ssdl|res://*/NorthwindModel.msl;
    provider=System.Data.SqlClient;
    provider connection string="data source=localhost;
    initial catalog=Northwind;
    integrated security=True;
    MultipleActiveResultSets=True;
    App=EntityFramework";"
    providerName="System.Data.EntityClient" />
</connectionStrings>
```

“数据库优先 (DataBase First)”
开发模式

Entity Framework的开发模式

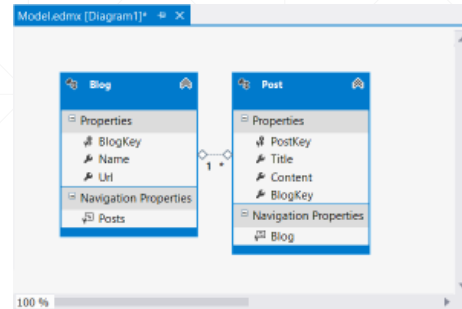


Code
First

Database
First

Model
First

使用可视化设计器



使用纯代码

```
public class Blog
{
    public int BlogKey { get; set; }
    public string Name { get; set; }
    public string Uri { get; set; }

    public virtual List<Post> Posts {
    }
}

public class Post
```

没有
数据库

模型优先 (Model First)

在EF设计器中创建数据模型
依据数据模型创建数据库
依据数据模型自动生成代码

代码优先 (Code First)

使用代码定义数据实体类以及它们之间的关联
使用代码创建数据库
当实体类有改变时，使用数据迁移更新数据库

已有
数据库

数据库优先 (Database First)

依据数据库架构生成可视化模型
依据可视化模型生成实体类代码

代码优先 (Code First)

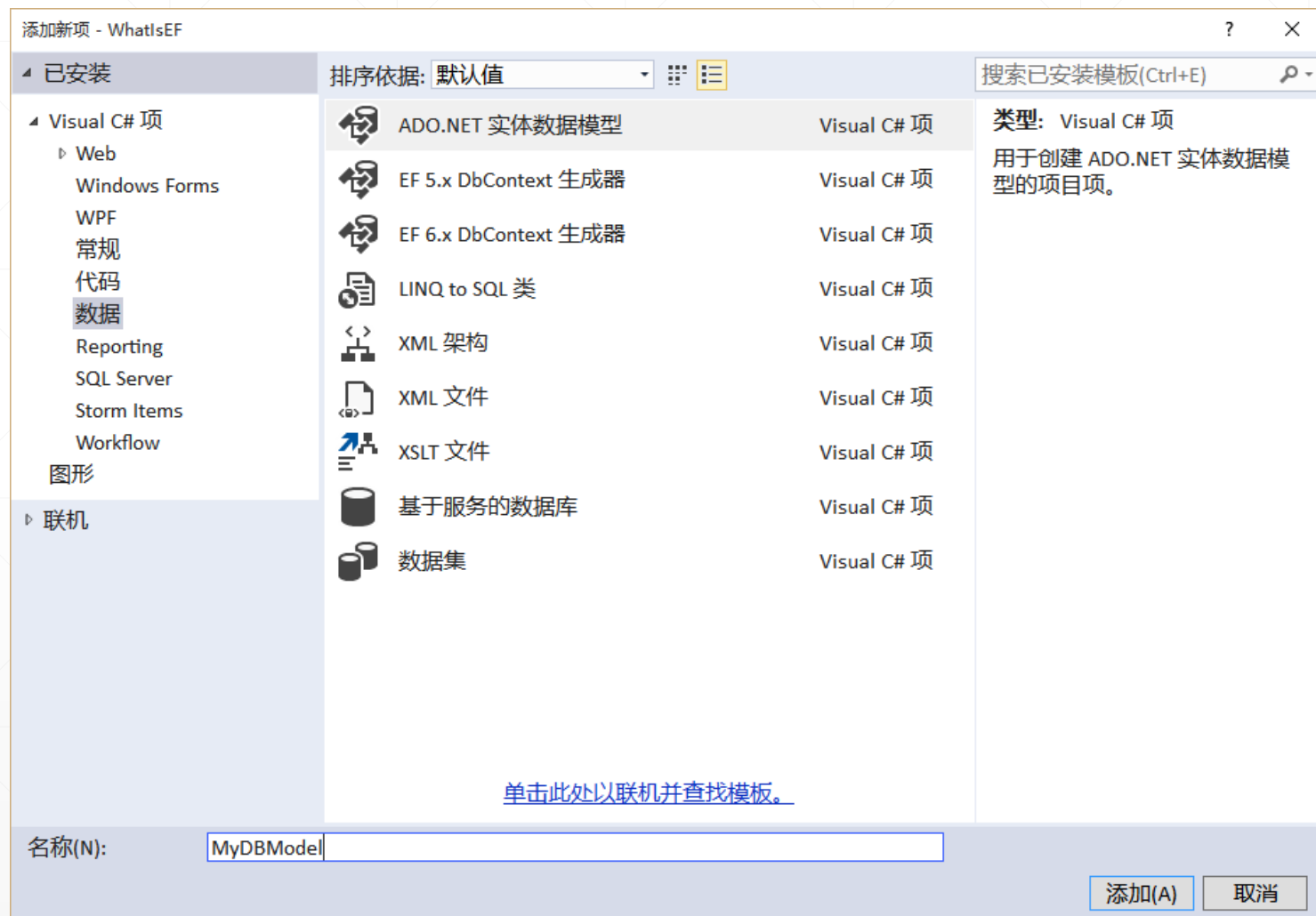
使用EF Power Tools依据数据库架构直接生成数据实体类

3 Database First开发模式

Database First开发模式概述

- 1 依据数据库架构生成可视化模型
- 2 依据可视化模型，基于代码模板生成数据实体类代码

向项目中添加“ADO.NET 实体数据模型”





选择模型内容

模型将包含哪些内容?(W)

来自数据库的
EF 设计器空 EF 设计器
模型空 Code First
模型来自数据库的
Code First

基于现有数据库在 EF 设计器中创建一个模型。您可以选择数据库连接、模型设置以及要在模型中包括的数据库对象。从该模型生成您的应用程序将与之交互的类。

< 上一步(P)

下一步(N) >

完成(F)

取消

实体数据模型向导

选择您的数据连接

您的应用程序应使用哪个数据连接与数据库进行连接?(w)

desktop-vldjaa3\sqlexpress.MyDB.dbo1 新建连接(C)...

此连接字符串似乎包含连接数据库所需的敏感数据(例如密码)。在连接字符串中存储敏感数据可能有安全风险。是否要在连接字符串中加入这些敏感数据?

☐ 否, 从连接字符串中排除敏感数据。我将在应用程序代码中设置此数据。(E)

☐ 是, 在连接字符串中包括敏感数据。(I)

连接字符串:

```
metadata=res:/**/*.MyDBModel.csdl|res:/**/*.MyDBModel.ssdl|
res:/**/*.MyDBModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=localhost\sqlexpress;initial catalog=MyDB;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ 将 App.Config 中的连接设置另存为(S):

MyDBContext

< 上一步(P) 下一步(N) > 完成(F) 取消

这个同时也是生成的
DbContext子类名字

连接属性

输入信息以连接到选定的数据源, 或单击“更改”选择另一个数据源和/或提供程序。

数据源(S):

Microsoft SQL Server (SqlClient) 更改(C)...

服务器名(E):

localhost\sqlexpress 刷新(R)

登录到服务器

☒ 使用 Windows 身份验证(W)

☐ 使用 SQL Server 身份验证(Q)

用户名(U):

密码(P):

☐ 保存密码(s)

连接到数据库

☒ 选择或输入数据库名称(D):

MyDB

☐ 附加数据库文件(H):


浏览(B)...

逻辑名(L):


高级(V)...


测试连接(T) 确定 取消


实体数据模型向导


选择您的数据库对象和设置


您要在模型中包括哪些数据库对象?(w)


▼ ☒  dbo


☒  Book


☒  BookCategory


☒  BookReview


☐  Concurrency


☐  EnumTest


☐  IdentityCard


☐  MyClass

☐  MyClassChild

☐  MyClassContainer

☐  OnlyTest

☒  OrderClient

☐  OrderClientBackup

☒ 确定所生成对象名称的单复数形式(s)

☒ 在模型中包括外键列(k)

☒ 将所选存储过程和函数导入到实体模型中(l)

模型命名空间(M):

< 上一步(P)

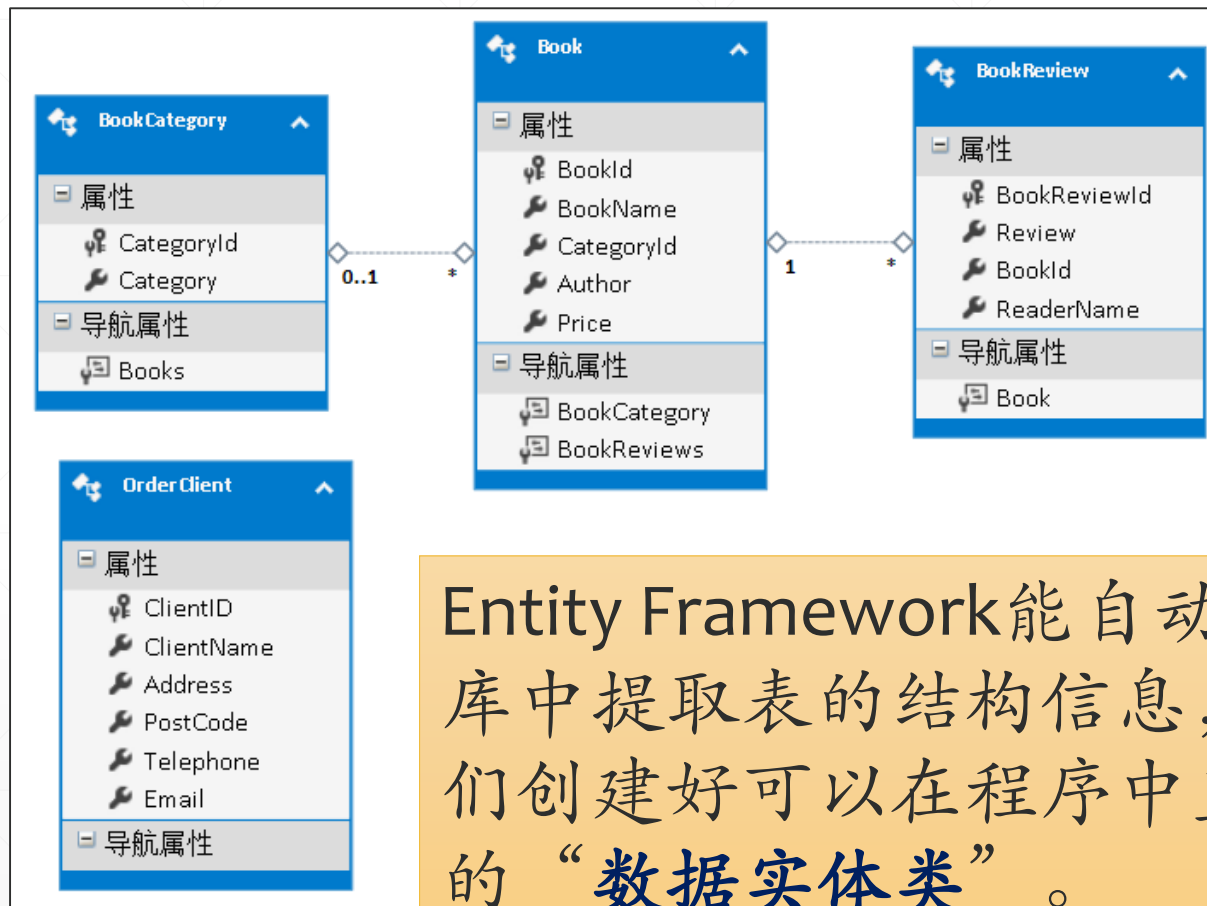
下一步(N) >

完成(F)

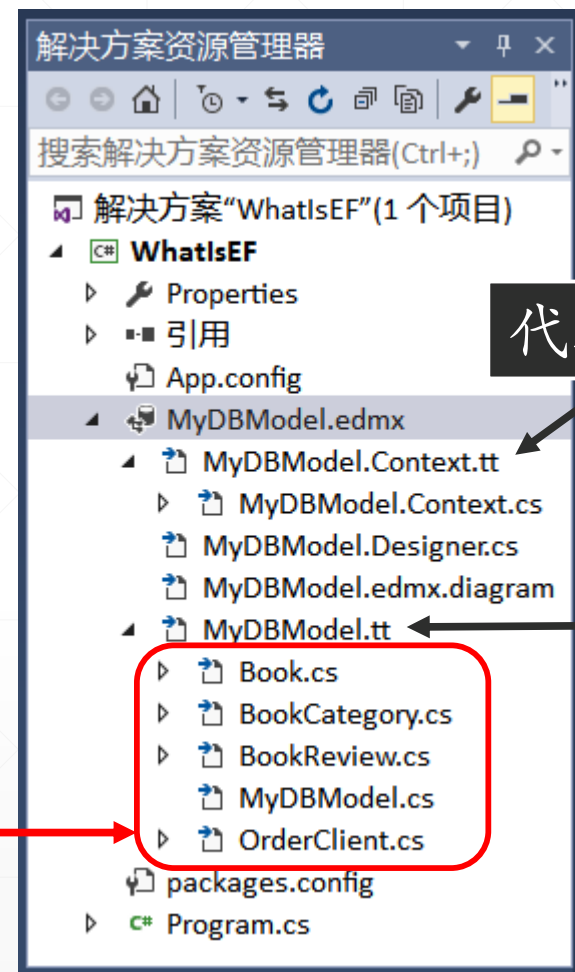
取消

建议选中这项

实体对象模型的生成



Entity Framework能自动地从数据库中提取表的结构信息，帮助我们创建好可以在程序中直接使用的“数据实体类”。



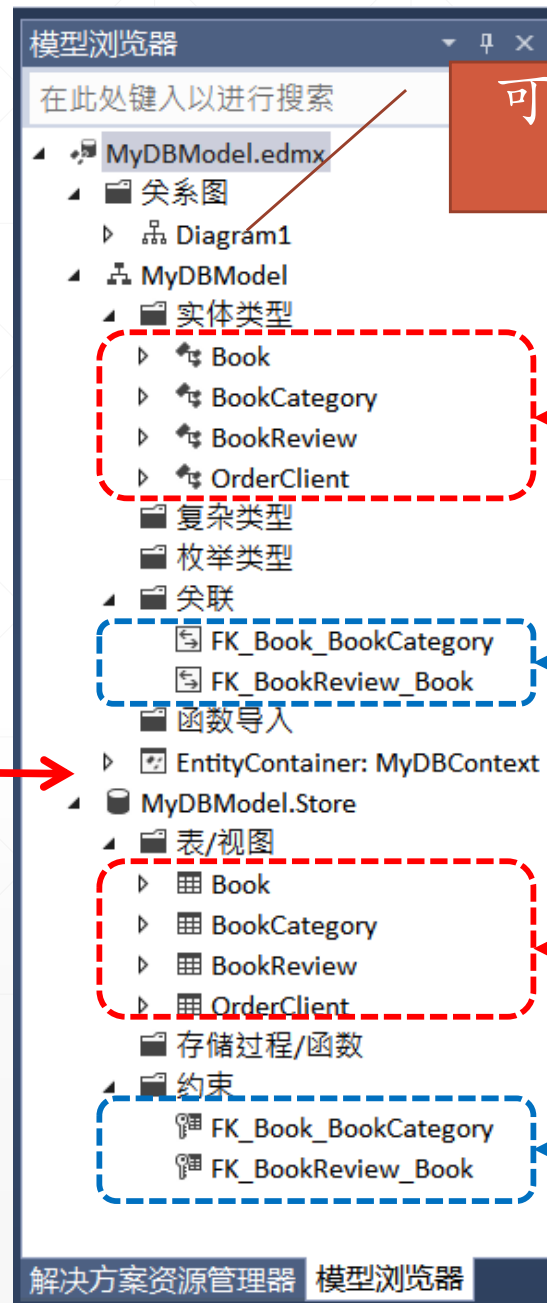
代码模板

查看所有实体对象模型



右击可视化设计器

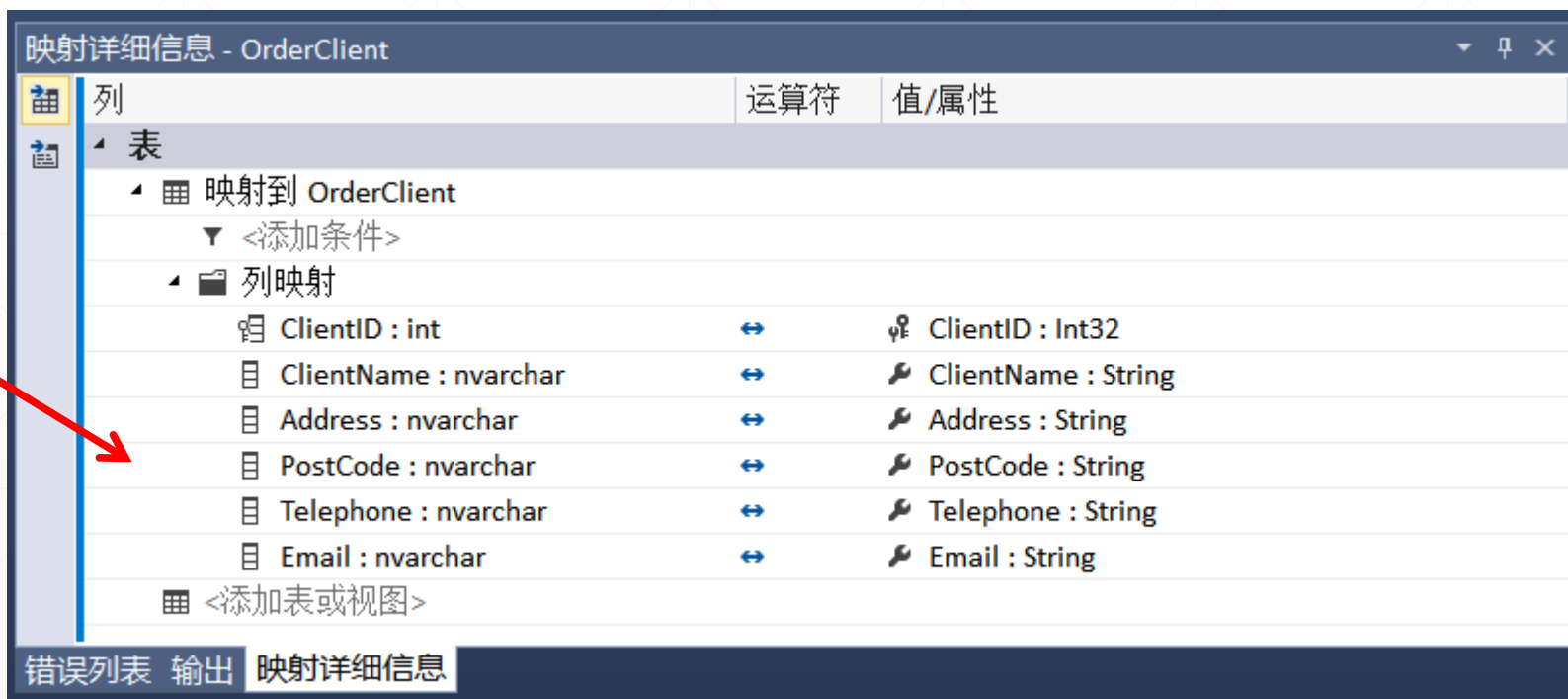
可以在模型浏览器中直接修改实体对象模型，其方法是在相应节点上右击，从弹出菜单中选择相应的命令。



可以创建多个关系图

查看映射模型

可以在“映射详细信息”面板中直接调整表
和对象属性之间的对应关系，或者将多个表
中的字段映射到同一个对象的不同属性。



如果数据库结构变更...



为什么要创建一个Entity Framework的实体对象模型，是为了以面向对象的方式实现数据的增、删、改、查.....

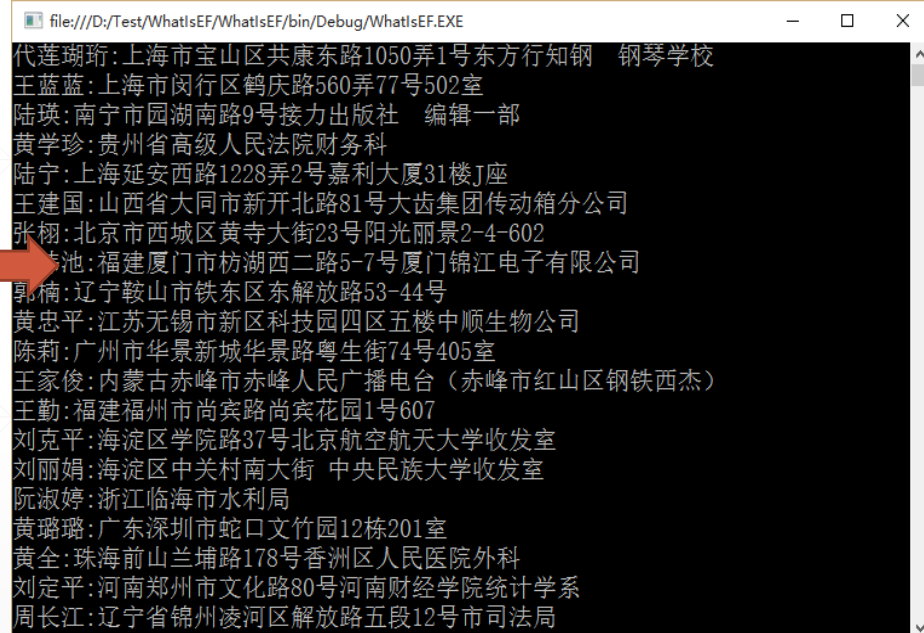
通过EF查询数据

```
private static void FetchData()
{
    var context = new MyDBContext();

    //使用LINQ to Entities查询数据
    var query = from client in context.OrderClients
                select client;

    //循环遍历结果集
    foreach (var client in query)
    {
        Console.WriteLine("{0}:{1}",
                           client.ClientName, client.Address);
    }

    //关闭数据库连接
    context.Dispose();
}
```



file:///D:/Test/WhatIsEF/WhatIsEF/bin/Debug/WhatIsEF.EXE

代莲璐琦:上海市宝山区共康东路1050弄1号东方行知钢 钢琴学校
王蓝蓝:上海市闵行区鹤庆路560弄77号502室
陆瑛:南宁市园湖南路9号接力出版社 编辑一部
黄学珍:贵州省高级人民法院财务科
陆宁:上海延安西路1228弄2号嘉利大厦31楼J座
王建国:山西省大同市新开北路81号大齿集团传动箱分公司
张翔:北京市西城区黄寺大街23号阳光丽景2-4-602
陈池:福建厦门市枋湖西二路5-7号厦门锦江电子有限公司
郭楠:辽宁鞍山市铁东区东解放路53-44号
黄忠平:江苏无锡市新区科技园四区五楼中顺生物公司
陈莉:广州市华景新城华景路粤生街74号405室
王家俊:内蒙古赤峰市赤峰人民广播电台 (赤峰市红山区钢铁西杰)
王勤:福建福州市尚宾路尚宾花园1号607
刘克平:海淀区学院路37号北京航空航天大学收发室
刘丽娟:海淀区中关村南大街 中央民族大学收发室
阮淑婷:浙江临海市水利局
黄璐璐:广东深圳市蛇口文竹园12栋201室
黄全:珠海前山兰埔路178号香洲区人民医院外科
刘定平:河南郑州市文化路80号河南财经学院统计学系
周长江:辽宁省锦州凌河区解放路五段12号市司法局

Entity Framework的**DbContext**类实现了**IDisposable**接口，在其**Dispose()**方法中关闭数据库连接。

```
public interface IDisposable {  
    void Dispose();  
}
```

当一个类型实现了IDisposable接口，说明这个类需要使用一些比较特殊的资源（比如文件句柄和数据库连接），这些资源因为数目有限，用完后必须及时归还给操作系统。

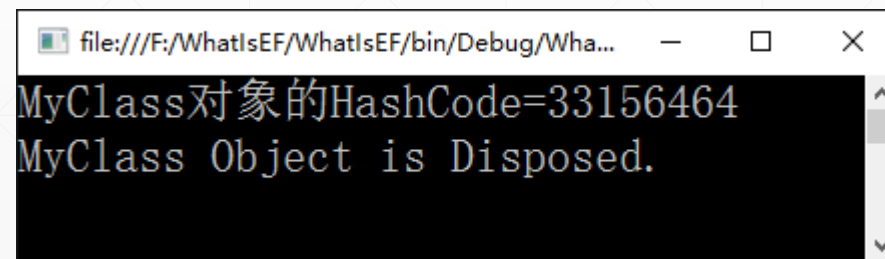
IDisposable接口所定义的**Dispose()**方法负责向操作系统归还占用的资源。

using关键字与IDisposable 接口

C#中定义了一个**using**关键字，可以很方便地自动调用实现了IDisposable接口的对象的Dispose()方法，无需手工显式调用。

```
class MyClass : IDisposable
{
    1个引用
    public void printMessage()
    {
        Console.WriteLine("MyClass对象的HashCode={0}",
            this.GetHashCode());
    }
    1个引用
    public void Dispose()
    {
        Console.WriteLine("MyClass Object is Disposed.");
    }
}
```

```
private static void UnderstandUsingKeyword()
{
    using(var obj=new MyClass())
    {
        obj.printMessage();
    }
}
```



The screenshot shows a console window with the following output:

```
MyClass对象的HashCode=33156464
MyClass Object is Disposed.
```

揭秘C#中的using关键字

```
using (MyClass obj = new MyClass())  
{  
    //使用obj干活……  
}
```



```
MyClass obj = new MyClass();  
  
try {  
    //使用obj干活……  
}  
finally {  
    IDisposable disposable = obj as IDisposable;  
    if (obj != null)  
        obj.Dispose(); //释放资源  
}
```

使用using关键字时，它所关联的对象必须实现IDisposable接口

使用using关键字让EF自动关闭数据库连接

```
private static void FetchDataUseUsingKeyword()
{
    //using结构所监控的对象，必须实现IDisposable接口
    using (var context = new MyDbContext())
    {
        //使用LINQ to Entities查询数据
        var query = from client in context.OrderClients
                     select client;
        //循环遍历结果集
        foreach (var client in query)
        {
            Console.WriteLine("{0}:{1}", client.ClientName,
                               client.Address);
        }
    }
}
```

数据库连接是一种宝贵的系统资源，除非是单用户的桌面应用，否则，都应该遵循“**即用即连**”，“**不用则断**”的方式，尽量减少长期占用一个数据库连接不释放的场景出现。

利用using关键字，有助于保证在各种情况下（正常运行和出现异常情况），DbContext的Dispose()方法都会被调用，其打开的数据库连接会被及时关闭。

Entity Framework如何从数据库中提取数据?

创建数据库
连接对象

连接上数
据库

发送SQL
命令，提
取数据

关闭数据
库连接

```
private static void HowLINQIsWorking()
{
    //using结构所监控的对象, 必须实现IDisposable接口
    using (var context = new MyDBContext())
    {
        //指示要记录EF所进行的所有数据库相关活动
        //Log属性是一个Action<String>委托
        //我们可以把任意一个符合此委托的方法传给它,
        //从而以自己的方式处理这些数据库活动记录
        //本例中使用Console.WriteLine方法,
        //表明希望在控制台窗口中输出这些信息
        context.Database.Log = Console.WriteLine;
        //使用LINQ to Entities查询数据
        var query = from client in context.OrderClients
                    select client;
        //循环遍历结果集
        foreach (var client in query)
        {
            Console.WriteLine("{0}:{1}", client.ClientName,
                               client.Address);
        }
    }
}
```



```
file:///D:/Test/WhatIsEF/WhatIsEF/bin/Debug/WhatIsEF.EXE
Opened connection at 2015/11/29 17:25:48 +08:00

SELECT
    [Extent1].[ClientID] AS [ClientID],
    [Extent1].[ClientName] AS [ClientName],
    [Extent1].[Address] AS [Address],
    [Extent1].[PostCode] AS [PostCode],
    [Extent1].[Telephone] AS [Telephone],
    [Extent1].[Email] AS [Email]
FROM [dbo].[OrderClient] AS [Extent1]

-- Executing at 2015/11/29 17:25:49 +08:00
-- Completed in 9 ms with result: SqlDataReader
```

LINQ查询实际上会被Entity Framework转换为SQL命令再发给数据库, 通过设置断点单步执行, 可以看到直到foreach开始, SQL命令才发送到SQL Server数据库。

小结：SQL命令的发送时机

仅仅访问DbSet属性不会执行查询，只有当以下情景发生时，查询才真正地发送到数据库：

1

使用foreach访问LINQ查询变量

2

调用了ToArray、ToList或ToDictionary方法

3

使用First或Any等标准查询扩展方法

4

调用了DbSet的Load, DbSet的Reload或Database的ExecuteSqlCommand方法

补充：EF 对象模型与数据库间的映射

关系数据库的世界	数据库应用程序的世界
数据库	DbContext类
表	DbContext中的DbSet<实体类名>
表间的关联	实体类之间的关联
表中的字段	实体类的公有属性
表中的单条记录	单个实体类的对象
视图	DbContext中的DbSet<视图名称>
存储过程	DbContext中的公有方法