# EntityFramework实体框架入门

## EntityFramework实体框架简介

1. 参考百科介绍
2. Visual Studio 2019/.NET Framework 4.8/Entity Framework 6
3. <https://wenku.baidu.com/view/6cc066a943323968011c92ca.html>
4. <https://baike.sogou.com/v22707385.htm?fromTitle=Entity+Framework>
5. <https://docs.microsoft.com/en-us/ef/>
6. 课堂源码：<https://github.com/zmrbak/EntityFramework>

## C#中委托的使用

1. C#委托可以避免在程序中大量实用if-else语句，同时使程序具有更好的扩展性。
2. 委托使一种引用类型（类似于class），它表示对具有特定参数列表和返回类型的方法的引用。
3. 现实生活中：中介（委托）。
4. 申明委托

public delegate 返回值 TestDelegate(参数列表);

public delegate string TestDelegate(string name);

1. 使用委托

//实例化委托

TestDelegate testDelegate1 = new TestDelegate(Test1);

string a=testDelegate1("zqm");

static string Test1(string name)

{

return "Hello,"+name;

}

1. 多播委托

+=添加委托

-=删除委托

-=删除不存在的委托时，不报错。

1. 使用委托将方法作为参数传递给另一个方法

public class User

{

public delegate void UserAction();

public void TackAction(UserAction userAction)

{

//userAction.Invoke();

userAction();

}

}

User user = new User();

user.TackAction(Action1);

user.TackAction(Action2);

static void Action1()

{

Console.WriteLine("Action1");

}

## C#中隐式类型与匿名类型的使用

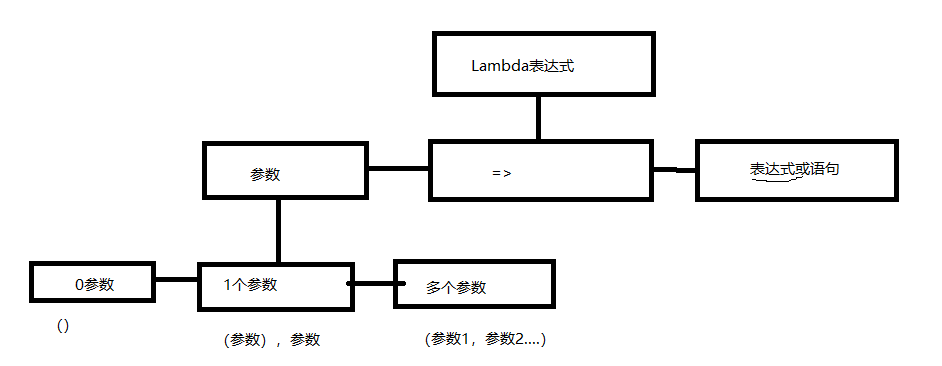
## C#中扩展方法与yield关键字

扩展方法的目的：为现有的类型添加一个方法。

## Lambda表达式

=>

Goes to，转到，成为



简化编写的代码量

让程序看起来更简洁

Action是一个委托类型

## LINQ查询

查询语句：从一种数据源检索数据的表达式。

SQL

XQuery

LINQ：跨数据源和数据格式的统一模型，实现查询。

XML、SQL数据库、ADO.NET数据集、.NET集合中的数据，任何数据格式。

## LINQ方法（1）

LINQ方法：非常灵活，本质上是通过扩展方法和Lambda表达式来创建查询。

LINQ查询，两种语法:

1. 方法语法（Fluent Syntax）,简化代码
2. 查询语法（Query Syntax）

## LINQ方法（2）

## LINQ操作

1. 聚合操作：从值集合中计算单个值。总和、最大值、最小值、平均值、元素数量。

Count

LongCount

Max

Min

Average

Sum

1. 元素操作：从一个序列中返回一个元素。

ElementAt

ElementAtOrDefault

First

FirstOrDefault

Last

LastOrDefault

Single

SingleOrDefault

## 数据类型转换（1）

ToList

ToArray

ToDictionary

ToHashSet

ToLookup

AsEnumerable

AsQueryable

AsParallel

Cast

OfType

## 数据类型转换（2）

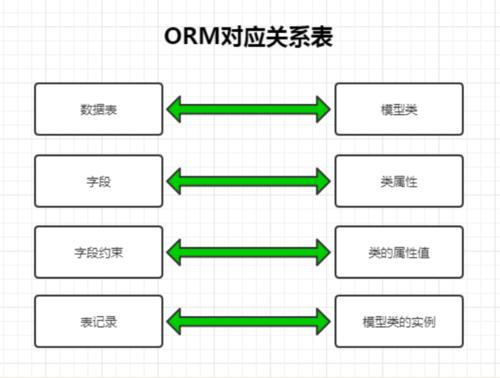
## 什么是ORM技术

O：Object 对象 ->类

R：Relational关系 ->数据库中的表

M：Mapping 映射 ->对应关系

ORM：翻译为：对象关系映射，也就是说将关系数据库中的数据用对象的形式表现出来，并通过面向对象的方式将这些对象组织起来，实现业务逻辑的过程。



LINQ，是一套通用的查询语法。

Entity Framework/EF，ADO.NET实体数据框架，

SQL Server、Oracle、MySQL、SQLite。。。

EF：将C#代码转换为SQL语句，实现对数据库和表的操作。

无ORM：

1. 操作数据库代码的重复性。

打开数据（连接数据库）

准备SQL语句

查询

读取查询结果

关闭数据库

SQLHelper类

增删改查

ORM，自动生成这些操作代码。

1. 大量SQL语句影响程序的扩展性和灵活性

SQL Server

SqlConnection

SqlCommand

……

Oracle

OracleConnection

OracleCommand

。。。。。。

SQLite

MySQL

不同数据库中SQL语法的个体差异。

ORM：编写一次代码，即可用于多种不同数据库。ORM自动针对不同数据库编写特定的SQL语句。

ORM：业务实体，在数据库中表现为关系数据，而在内存中则表现为对象。

编码擅长处理对象，很难处理关系数据。

ORM通过关系数据与对象数据之间的映射，自动产生SQL语句。

Entity Framework：

1. 支持多种数据
2. 映射引擎支持存储过程
3. VS集成，可视化操作。

## EF“Database First”建模（1）

Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Zmrbak\source\repos\EF13\EF13\EF13.mdf;Integrated Security=True;Connect Timeout=30

SSDL：存储模型，指数据库中的实体（将各个实体的关系固定到表中的形式）。

CSDL：概念模型，以程序语言的角度来定义模型，即其中定义的实体、主键、属性、关联等都是对应于.NET Framework中的类型。

C-S mapping：映射，将概念模型和存储模型连接起来，以便执行操作。

实体.属性（概念模型）🡪 实体.字段（存储模型）

## EF“Database First”建模（2）

## 使用EF完成CRUD（1）

C：Create，增加 INSERT

Add(sampleTable);

R：Retrieve，检索 SELECT

Find(3);

U：Update，更新 UPDATE

temp.Name = "李四";

D：Delete，删除 DELETE

Remove(temp);

eF13Entities.SaveChanges();

增删改查

App.config

<configSections>

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

</configSections>

<connectionStrings>

<add name="EF13Entities" connectionString="metadata=res://\*/Model1.csdl|res://\*/Model1.ssdl|res://\*/Model1.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=(LocalDB)\MSSQLLocalDB;attachdbfilename=|DataDirectory|\EF13.mdf;integrated security=True;connect timeout=30;MultipleActiveResultSets=True;App=EntityFramework&quot;"

providerName="System.Data.EntityClient" />

</connectionStrings>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">

<parameters>

<parameter value="mssqllocaldb" />

</parameters>

</defaultConnectionFactory>

<providers>

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

</providers>

</entityFramework>

## 使用EF完成CRUD（2）

## 在EF中执行SQL语句：ExecuteSqlCommand

## 在EF中执行SQL语句：SqlQuery（1）

## 在EF中执行SQL语句：SqlQuery（2）

## 在EF中执行存储过程（1）

## 在EF中执行存储过程（2）

## 在EF中进行多表CRUD（1）

主键

外键

## 在EF中进行多表CRUD（2）

## EF中的Code First模式

领域驱动设计原则

## Code First模式中的多表创建（1）

## Code First模式中的多表创建（2）

## Code First模式中的模型重建

## 空EF设计器模型

## 来自数据库的Code First模型

看相关书籍

不建议读百度，知识太零散。

《EntityFramework实体框架进阶》

源代码 <https://github.com/dotnet/ef6>